ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Sistema de Autenticación Accesible para el Hogar Digital

AUTOR: Gonzalo Antón García

TITULACIÓN: Grado en Ingeniería Telemática

TUTOR: Miguel Ángel Valero Duboy

DEPARTAMENTO: Dpto. de Ingeniería y Arquitecturas Telemáticas

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Jesús Arriaga García de Andoaín

VOCAL: Ana Gómez Oliva

SECRETARIO: Miguel Ángel Valero Duboy

Fecha de lectura:

Calificación:

El Secretario,

Resumen

El Hogar Digital Accesible (HDA) de la ETSIST nace con el propósito de acercar las nuevas Tecnologías de la Información a las personas que precisan de necesidades concretas de accesibilidad y usabilidad, dotándoles de herramientas que les permitan aumentar su calidad de vida, confort, seguridad y autonomía. El entorno del HDA consta de elementos de control para puertas, persianas, iluminación, agua o gas, sensores de temperatura, incendios, gas, sistemas de climatización, sistemas de entretenimiento y sistemas de seguridad tales como detectores de presencia y alarmas. Todo ello apoyado sobre una arquitectura de red que proporciona una pasarela residencial y un acceso a banda ancha.

El objetivo principal de este PFG ha sido el desarrollo de un sistema de autenticación para el Hogar Digital Accesible de bajo coste. La idea de integrar un sistema de autenticación en el HDA, surge de la necesidad de proteger de accesos no deseados determinados servicios disponibles dentro de un ámbito privado. Algunos de estos servicios pueden ser tales como el acceso a la lectura de los mensajes disponibles en el contestador automático, el uso de equipos multimedia, la desconexión de alarmas de seguridad o simplemente la configuración de ambientes según el usuario que esté autenticado (intensidad de luz, temperatura de la sala, etc.).

En el desarrollo han primado los principios de accesibilidad, usabilidad y seguridad necesarios para la creación de un entorno no invasivo, que permitiera acreditar la identidad del usuario frente al sistema HDA.

Se ha planteado como posible solución, un sistema basado en el reconocimiento de un trazo realizado por el usuario. Este trazo se usará como clave de cara a validar a los usuarios. El usuario deberá repetir el trazado que registró en el sistema para autenticarse. Durante la ejecución del presente PFG, se justificará la elección de este mecanismo de autenticación frente a otras alternativas disponibles en el mercado.

Para probar la aplicación, se ha podido contar con dos periféricos de distintas gamas, el uDraw creado para la PS3 que se compone de una tableta digitalizadora y un lápiz que permite recoger los trazos realizados por el usuario de forma inalámbrica y la tableta digitalizadora Bamboo de Wacom. La herramienta desarrollada permite a su vez, la posibilidad de ser usada por otro tipo de dispositivos como es el caso del reloj con acelerómetro de 3 ejes de Texas Instruments Chronos eZ430 capaz de trasladar los movimientos del usuario al puntero de un ratón.

El PFG se encuentra dividido en tres grandes bloques de flujo de trabajo. El primero se centra en el análisis del sistema y las tecnologías que lo componen, incluyendo los distintos algoritmos disponibles para realizar la autenticación basada en reconocimiento de patrones aplicados a imágenes que mejor se adaptan a las necesidades del usuario. En el segundo bloque se recoge una versión de prueba basada en el análisis y el diseño UML realizado previamente, sobre la que se efectuaron pruebas de concepto y se comprobó la viabilidad del proyecto. El último bloque incluye la verificación y validación del sistema mediante pruebas que certifican que se han alcanzado los niveles

de calidad necesarios para la consecución de los objetivos planteados, generando finalmente la documentación necesaria.

Como resultado del trabajo realizado, se ha obtenido un sistema que plantea una arquitectura fácilmente ampliable lograda a través del uso de técnicas como la introspección, que permiten separar la lógica de la capa de negocio del código que la implementa, pudiendo de forma simple e intuitiva sustituir código mediante ficheros de configuración, lo que hace que el sistema sea flexible y escalable.

Tras la realización del PFG, se puede concluir que el producto final obtenido ha respondido de forma satisfactoria alcanzando los niveles de calidad requeridos, siendo capaz de proporcionar un sistema de autenticación alternativo a los convencionales, manteniendo unas cotas de seguridad elevadas y haciendo de la accesibilidad y el precio sus características más reseñables.

Abstract

Accessible Digital Home (HDA) of the ETSIST was created with the aim of bringing the latest information and communications technologies closer to the people who has special needs of accessibility and usability increasing their quality of life, comfort, security and autonomy. The HDA environment has different control elements for doors, blinds, lighting, water or gas, temperature sensors, fire protection systems, gas flashover, air conditioning systems, entertainments systems and security systems such as intruders detectors and alarms. Everything supported by an architecture net which provides a broadband residential services gateway.

The main goal of this PFG was the development of a low-cost authentication system for the Accessible Digital Home. The idea of integrating an authentication system on the HDA, stems from the need to safeguard certain private key network resources from unauthorized access. Some of said resources are the access to the answering machine messages, the use of multimedia devices, the alarms deactivation or the parameter settings for each environment as programmed by the authenticated user (light intensity, room temperature, etc.).

During the development priority was given to concepts like accessibility, usability and security. All of them necessary to create a non invasive environment that allows the users to certify their identity.

A system based on stroke pattern recognition, was considered as a possible solution. This stroke is used as a key to validate users. The user must repeat the stroke that was saved on the system to validate access. The selection of this authentication mechanism among the others available options will be justified during this PFG.

Two peripherals with different ranges were used to test the application. One of them was uDraw design for the PS3. It is wireless and is formed by a pen and a drawing tablet that allow us to register the different strokes drawn by the user. The other one was the Wacom Bamboo tablet, that supports the same functionality but with better accuracy. The developed tool allows another kind of peripherals like the 3-axes accelerometer digital wristwatch Texas Instruments Chronos eZ430 capable of transfering user movements to the mouse cursor.

The PFG is divided by three big blocks that represent different workflows. The first block is focused on the system analysis and the technologies related to it, including algorithms for image pattern recognition that fits the user's needs. The second block describes how the beta version was developed based on the UML analysis and design previously done. It was tested and the viability of the project was verified. The last block contains the system verification and validation. These processes certify that the requirements have been fulfilled as well as the quality levels needed to reach the planned goals. Finally all the documentation has been produced.

As a result of the work, an expandable system has been created, due to the introspection that provides the opportunity to separate the business logic from the code that implements it. With this technique, the code could be replaced throughout configuration files which makes the system flexible and highly scalable.

Once the PFG has finished, it must therefore be concluded that the final product has been a success and high levels of quality have been achieved. This authentication tool gives us a low-cost alternative to the conventional ones. The new authentication system remains security levels reasonably high giving particular emphasis to the accessibility and the price.

Índice de Contenidos

Tabla de Ilustraciones

Lista de Acrónimos

- **BIOS:** Basic Input-Output System
- **CMMI:** Capability Maturity Model Integration
- **DAO:** Data Access Object
- **DCT:** Discrete Cosine Transform
- **DOM:** Document Object Model
- **GUI:** Graphical User Interface
- **HDA:** Hogar Digital Accesible.
- **JVM:** Java Virtual Machine
- **MPEG:** Moving Picture Experts Group
- **PIN:** Personal Identification Number
- **PKI:** Public Key Infrastructure
- **PPQA:** Product and Process Quality Asurance
- **QA:** Quality Asurance
- **RSA:** Algoritmo de cifrado creado por Rivest, Shamir Adleman
- **RUP:** Rational Unified Process
- **SAX:** Simple API for XML
- **UID:** Unique Idenfiers
- **UML:** Unified Modeling Language
- **XML:** Extensible Markup Language
- **XSD:** XML Schema Definition

Capítulo 1: Introducción

1.1. Contextualización

La tecnología se encuentra presente en todos los ámbitos de nuestra vida. Si por un momento nos paramos a pensar en cualquier actividad que desempeñamos a diario, podremos encontrar buenos ejemplos de elementos tecnológicos que nos acompañan en el transcurso del día. Desde el zumbido del despertador por la mañana, hasta la última luz que apagamos antes de ir a dormir, la tecnología nos acompaña, de forma sutil y discreta, pero inherente a nuestro estilo de vida.

Cuando la tecnología entra en escena en el ámbito doméstico y *"a través de equipos y sistemas, y la integración tecnológica entre ellos, ofrece a sus habitantes funciones y servicios que facilitan la gestión y el mantenimiento del hogar, aumentan la seguridad; incrementan el confort; mejoran las telecomunicaciones; ahorran energía, costes y tiempo, y ofrecen nuevas formas de entretenimiento, ocio y otros servicios dentro de la misma y su entorno"*, tenemos lo que se conoce como Hogar Digital (1)

El Hogar Digital, nace como concepto a finales de los 70 en concreto en 1975 de la mano del protocolo de comunicaciones X10. La sencillez y la accesibilidad al protocolo hicieron que rápidamente emergieran multitud de aplicaciones que serían instaladas en grandes promociones inmobiliarias (2). Este protocolo sigue vivo en la actualidad y en pleno auge, debido a que su implantación es simple y el coste es asumible. No obstante en el mercado han ido apareciendo paulatinamente soluciones más sofisticadas que han ido completando el mercado de los sistemas domóticos.

A finales de los 90 tres empresas, EIB, Batibus y EHS, se fusionan para dar lugar a uno de los estándares abiertos más extendidos en Europa, el KNX (3). En 2006 KNX Association emitía una nota de prensa informando al mundo que el trámite para reconocer el protocolo como Estándar Internacional, había sido aprobado por los distintos organismos, surge así el **ISO/IEC 14543-3** (4).

Paralelamente en EEUU una empresa nacida en 1994 bajo el nombre de LonMark, fue ganando protagonismo con su plataforma de networking para aplicaciones domóticas LonWorks. Dicho sistema estaba basado en un protocolo propietario denominado LonTalk que con el tiempo llego a convertirse en el Estándar Internacional **ISO/IEC 14908-1** (5).

KNX, Lonworks y X10, son las principales y más relevantes plataformas sobre las que se asientan los sistemas de Hogar Digital actuales. Todas tienen en común la misma característica que consiste en tratar de mejorar nuestra calidad de vida proporcionando mayor confort, seguridad y entretenimiento.

Pero no siempre las tecnologías tienen un efecto positivo sobre nuestras vidas. Hay casos en los que estos avances no tienen repercusión sobre el individuo o bien las consecuencias del avance son negativas, al menos para determinados sectores de la población.

Un estudio realizado por el Instituto Nacional de Estadística en el año 1999, arrojaba datos contundentes sobre el número de personas con algún tipo de discapacidad en España, situándolo en el 9% de la población (6). Son datos muy elevados si nos paramos a pensar en el alcance que estos tienen.

La tendencia se ha mantenido en estudios posteriores, como es el caso del realizado en noviembre de 2008 bajo el acrónimo de EDAD (Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia) en la que el porcentaje disminuía en tan sólo un 0,5% con respecto al estudio realizado en 1999 (7).

La estabilidad en los porcentajes a lo largo de un periodo de 9 años, nos hace ser conscientes de que hay un gran sector de la población que demanda una serie de servicios adaptados a sus necesidades, que deben ser tenidos en cuenta a la hora de concebir el progreso tecnológico.

El informe EDAD nos proporciona unos datos que no podemos dejar pasar por alto. Nos habla de 608.000 personas que viven solas en su hogar con algún tipo de discapacidad, 1.390.000 personas que no pueden realizar alguna de las actividades básicas de la vida diaria sin ayuda o de 3.300.000 de hogares en los que reside una persona con discapacidad, esto es el 20% de los hogares españoles.

La evolución tecnológica, como tal, es un elemento neutro, no es ni buena, ni mala, todo depende del uso que se le dé. Contemplando las cifras expuestas por el INE, es fácil observar la necesidad de incluir en el desarrollo tecnológico a un sector de la población que abarca el 9%, evitando en la medida de lo posible, limitaciones en cuanto a la accesibilidad y usabilidad de las nuevas aplicaciones que vayan apareciendo.

El riesgo de exposición a la exclusión digital es todavía elevado en España. Según el informe eEspaña 2013 "*El factor que en España produce mayor exclusión digital es la edad de las personas, ya que los españoles de más de 55 años presentan niveles de exclusión digital más elevados que cualquier otro colectivo desfavorecido. Además, la exclusión digital de este segmento es mucho más grave en España que en el resto de la UE*" (8).

Ese riesgo de exclusión es aún más elevado si coexiste con algún tipo de discapacidad, en cuyo caso la no inclusión del colectivo en el avance tecnológico, está casi asegurada. Es frente a estas desigualdades, ante las que se deben tomar medidas de prevención y correctivas, para tratar de lograr un avance de la sociedad, colectivo, equitativo y justo para y con todos los sectores.

Con ese criterio en mente, nace el Hogar Digital Accesible (HDA) de la ETSIST, cuyo objetivo principal es precisamente, acercar las nuevas Tecnologías de la Información a las personas que precisen de necesidades concretas de accesibilidad y usabilidad, dotándoles de herramientas que les permitan aumentar su calidad de vida, confort, seguridad y autonomía (9).

El entorno del HDA consta de elementos de control para puertas, persianas, iluminación, agua o gas, sensores de temperatura, incendios, gas, sistemas de climatización, sistemas de entretenimiento y sistemas de seguridad tales como detectores de presencia y alarmas. Todo ello apoyado sobre una arquitectura de red que proporciona una pasarela residencial y un acceso a banda ancha.

El principal beneficio que pretende el HDA sobre los usuarios, es aumentar el grado de autonomía de éstos. Dotándoles de herramientas que aumenten la calidad de vida y el grado de confort e independencia.

Si bien el objetivo planteado ha sido ampliamente alcanzado en el HDA implementado por la ETSIST, con este proyecto se pretende completar el entorno, dotándole de un componente con capacidad de autenticar al usuario y así individualizar el comportamiento del sistema adaptándose a sus necesidades particulares.

Existen en la actualidad multitud de dispositivos con capacidad para determinar la identidad del usuario: reconocedores de huellas, de iris, reconocimiento de voz, reconocimiento de trazos, etc. De todos ellos se hablará en este PFG y se detallará el por qué de la elección de un sistema basado en reconocimiento de trazos para llevar a cabo su autenticación, aunque ya podemos avanzar que la principal razón es la búsqueda de un dispositivo de carácter no invasivo y barato con capacidad para reconocer movimientos que puedan ser trasladados a un puntero de ratón.

En este PFG se hablará de dos dispositivos, el uDraw de ps3 y un reloj de pulsera de Texas Instruments comercializado bajo el nombre de eZ430 Chronos dotado de acelerómetro de tres ejes con el que se puede controlar un puntero de ratón a distancia.

Ambos dispositivos utilizan tecnología de Radio Frecuencia para transmitir datos sin necesidad de hilos y son capaces de detectar el movimiento realizado por el usuario y trasladarlo a un equipo informático donde pueda ser analizado y procesado.

1.2. Objetivos del Proyecto

El objetivo principal de este proyecto es la realización de un sistema de autenticación para el Hogar Digital Accesible, basado en el principio de tecnología para las personas, que permita identificar al usuario de cara al sistema, de manera natural, no invasiva y con el menor coste posible. Se justificará la elección de un sistema basado en el dispositivo uDraw para PS3 y el reconocimiento de trazos realizados a mano, como una posible solución frente a otras opciones tecnológicas.

Para la consecución del objetivo principal, es necesario la realización de ciertos objetivos específicos, tales como:

- Estudio sobre los diferentes mecanismos autenticadores.
- Análisis de los distintos algoritmos existentes para la comparación de imágenes.
- Implementación de una prueba de concepto.
- Validación del sistema.
- Generación de documentación.

1.3. Estructura de la memoria

La memoria se encuentra dividida en 6 grandes secciones que a su vez se dividen en apartados más detallados.

La primera sección se corresponde con la *Introducción*, en ella se definen 3 apartados que proporcionan una visión general del conjunto de la memoria.

En la siguiente sección, titulada *Solución Propuesta,* se diferencian apartados para cada workflow del PFG siguiendo la metodología de procesos UP. En esta sección se puede encontrar la información más técnica de la memoria.

La tercera sección de la memoria, presenta los *Resultados* obtenidos al ejecutar de forma intensiva distintas pruebas concebidas siguiendo criterios del CMMI.

En la sección inmediatamente posterior, se pasan a obtener *Conclusiones* derivadas de analizar los resultados obtenidos en la sección anterior así como se procederá a validar si la solución propuesta ha sido la adecuada y las posibles mejoras que se puedan llevar a cabo en futuros desarrollos.

Las secciones quinta y sexta, contienen las *Referencias* consultadas para la elaboración de la solución así como un conjunto de *Anexos* donde poder consultar información complementaria como es un útil Manual de Usuario de recomendada lectura de cara a usar la aplicación.

Capítulo 2: Antecedentes y Análisis del Sistema

La metodología aplicada para el diseño, implementación y validación del sistema desarrollado en este PFG se corresponde con la definida en RUP (10). Como tal, el proceso se dividirá en cuatro fases sobre las que se iterará hasta completar el proceso.

Todo el proceso estará guiado por los Casos de Uso y centrado en la arquitectura. El procedimiento descrito en RUP debe garantizar que la solución sea desarrollada de forma iterativa e incremental.

En la siguiente figura, se muestran los procesos y fases definidos según RUP a la hora de afrontar un proyecto y la carga de trabajo que cada uno representa:

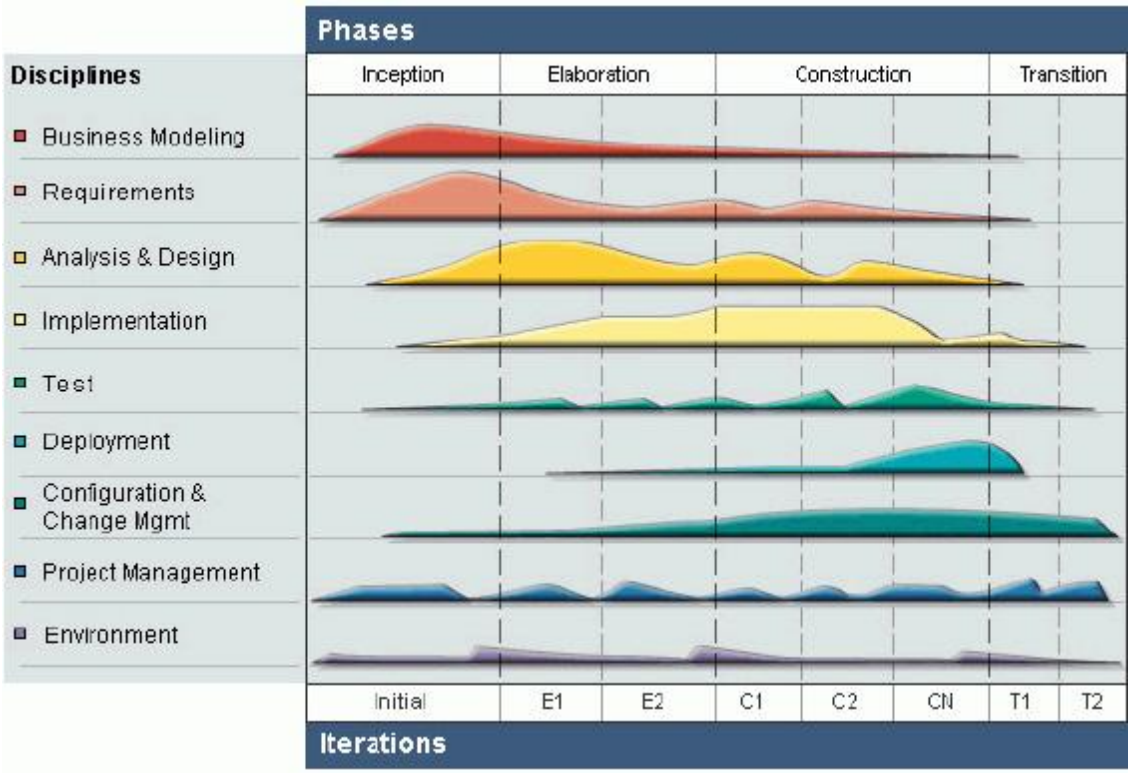


Ilustración 1: Fases Metodología RUP (Fuente: www.ibm.com)

2.1. Requisitos

Para la aceptación final del producto, el sistema objeto de este PFG debe cumplir con los siguientes criterios de aceptación, constituyendo estos lo que desde este momento se denominará el Alcance del Proyecto.

2.1.1. Requisitos Funcionales

El sistema:

- debe autenticar a un usuario de forma segura.
- debe generar un token que acredite la autenticación de cara a terceros.
- debe firmar el token proporcionando los servicios de integridad y no repudio en origen.
- debe permitir registrar nuevos trazos al usuario.
- debe permitir configurar el nivel de tolerancia con el que admitir un trazo.
- debe permitir cambiar la contraseña del Administrador.
- debe permitir seleccionar el periférico de entrada con el que realizar el trazo.
- debe permitir generar una nueva pareja de claves para la PKI.
- debe registrar los intentos de acceso en ficheros de log.
- debe permitir añadir nuevos algoritmos de comprobación de trazos.
- debe garantizar la persistencia de datos tras su apagado

2.1.2. Requisitos No Funcionales

El sistema:

- debe tener una GUI de usuario accesible.
- debe mantener la clave privada de la PKI cifrada.
- debe mantener la clave secreta del administrador cifrada.
- debe realizar la autenticación de forma rápida e intuitiva.
- debe ser robusto y estable.

2.1.3. Fuera de Alcance

Por tratarse de un sistema concebido con carácter de ámbito local, quedan fuera del alcance de este proyecto, los procesos y procedimientos relativos a la distribución y almacenamiento de claves.

2.2. Antecedentes tecnológicos

Para dar solución a las necesidades planteadas se han barajado varias opciones, de las cuales se ha seleccionado una que pasará a describirse a continuación

El objetivo principal del proyecto es la implementación de un sistema capaz de proporcionar un servicio de autenticación para un entorno de Hogar Digital Accesible, siendo preceptivo que éste sea del menor coste posible, no invasivo y que una de sus características esenciales sea la accesibilidad y simplicidad de uso.

Los sistemas que más se emplean para realizar mediciones biométricas son los basados en reconocimiento de trazo manuscrito, reconocimiento de gestos, reconocimiento de huellas dactilares, reconocimiento de iris y reconocimiento de voz.

Según criterios de RedIRIS (Red Académica y de Investigación Española) cuya finalidad es proporcionar servicios avanzados de comunicaciones a la comunidad científica y universitaria nacional, existen varios parámetros a tener en cuenta a la hora de seleccionar una u otra tecnología de reconocimiento de patrones biométricos.

La siguiente tabla obtenida de la web de RedIRIS (11) muestra los parámetros de cada tecnología, ponderados para que podamos compararlos de una forma sencilla y rápida:

Tabla 1: Sistemas Biométricos comparativa (Fuente: RedIRIS)

	IRIS	RETINA	HUELLA	MANO	FIRMA	VOZ
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta	Alta	Alta
Prevención de ataques	Muy Alta	Muy alta	Alta	Alta	Media	Media
Aceptación	Media	Media	Media	Alta	Muy alta	Alta
Estabilidad	Alta	Alta	Alta	Media	Media	Media
Identificación y autenticación	Ambas	Ambas	Ambas	Aut	Ambas	Aut
Estándares	-	-	ANSI/NIST, FBI	-	-	SVAPI
Interferencias	Gafas	Irritaciones	Suciedad, heridas, asperezas ...	Artritis, ...	Firmas fáciles o cambiantes	Ruido, resfriado ...
Utilización	Instalaciones nucleares, servicios médicos, centros penitenciarios	Instalaciones nucleares, servicios médicos, centros penitenciarios	Policía, industrial	General	Industrial	Accesos remotos en bancos o bases de datos
Precio por nodo en 1997(USD)	5000	5000	1200	2100	1000	1200

A pesar de que los precios están desactualizados, las relaciones entre ellos sigue siendo plenamente válida, pudiendo determinar que un dispositivo de reconocimiento de patrones basado en el iris es aproximadamente 5 veces más caro que uno basado en reconocimiento de firma.

Se han analizado los pros y contras de cada una de estas tecnologías para decidir cuál es la más apropiada para ser implementada como solución a la necesidad de un sistema de autenticación para el HDA.

Nuestro sistema está concebido desde su origen buscando una máxima, debe ser económico y ampliamente aceptado por los usuarios. La restricción económica, nos desecha de forma casi inmediata las tecnologías basadas en reconocimiento de iris y retina, ya que estadísticamente son las soluciones más caras del mercado.

Otra de las opciones que más se ajustaba, en cuanto a capacidades y precio, fue la basada en reconocimiento de huellas dactilares. En este caso, los motivos de no ser seleccionada estuvieron fundamentados en experiencias previas que evidenciaban una falta de precisión en cuanto a la validación. Asimismo, este sistema no tiene una aceptación muy alta por parte de los usuarios, siendo éstos reacios a incluir su uso entre sus actividades cotidianas.

Como consecuencia del análisis y dadas las particularidades del entorno en el cual se desea integrar nuestro sistema de autenticación, la solución que más se ajusta al planteamiento inicial, siguiendo como criterios principales de selección el precio y la aceptación por parte de los usuarios finales, es un sistema de reconocimiento de patrones basado en firma manuscrita.

Una de las principales fuentes que se han consultado para acreditar lo anteriormente expuesto, ha sido la Red Académica y de Investigación Española, RedIRIS de la cual ha sido extraída la tabla anterior (*Tabla 1*). Teniendo en cuenta el conjunto de parámetros que conforman nuestro entorno de aplicación, son estos los motivos que han determinado que la solución finalmente propuesta para implementar el sistema de autenticación que se va a integrar con el Hogar Digital Accesible (HDA), sea un sistema de reconocimiento de patrones biométricos basado en firma manuscrita.

Como periféricos de entrada para el reconocimiento de trazos, se pretende emplear la GameTablet uDraw para PS3 (12) y el reloj de Texas Instruments con acelerómetro de 3 ejes Chronos eZ430 (13), con el que se puede controlar un puntero de ratón a distancia.

Ambos dispositivos utilizan tecnología de Radio Frecuencia para transmitir datos sin necesidad de hilos y son capaces de detectar el movimiento realizado por el usuario y trasladarlo a un equipo informático donde pueda ser analizado y procesado. Son económicamente viables y no son dispositivos que provoquen rechazo en el usuario final. Estos dispositivos se han elegido principalmente por encontrarse entre los más económicos de su gama y por la facilidad de encontrar stock disponible. Cada uno de ellos cuenta con capacidad para el reconocimiento de trazos. No obstante, el proceso que siguen para registrar los movimientos efectuados por el usuario es distinto.

La GameTablet uDraw para PS3, es un dispositivo presentado en forma de tableta, con un lápiz que permite dibujar sobre ella. Cuenta con distintos botones inicialmente

concebidos para jugar con la PS3.La comunicación con el PC se realiza a través de RadioFrecuencia, empleando para ello la banda libre de los 2,4GHz. El dispositivo funciona mediante 3 pilas de tipo AA que proporcionan una autonomía un tanto escasa. La producción de este dispositivo ha sido descontinuada por parte de su fabricante THQ, no obstante, es fácil de encontrar stock disponible en grandes almacenes a un PVP desde 16,85€.

Para la conexión del uDraw con el PC ha sido necesario emplear el driver uDrawTabletv7.1.0, desarrollado por Brandon Wilson que es distribuido como open source y sin ningún tipo de restricción en cuanto a licencia como puede leerse en su web de descarga (14).

El Reloj de Texas Instruments eZ430 Chronos, es un dispositivo wireless basado en el chip CC430F6137 que hace uso de la banda de los 433MHz. El dispositivo viene integrado con sensores de temperatura y presión, así como con un acelerómetro de 3 ejes, capaz de registrar el movimiento que se realiza con él. El dispositivo es reprogramable a través de una completa suite del fabricante. La idea es usar el acelerómetro de 3 ejes del reloj para transmitir el movimiento al PC y controlar el puntero del ratón para analizar y procesar los movimientos realizados por el usuario. El dispositivo se encuentra a la venta en la página oficial del fabricante a un precio de 58$.

La tecnología que residirá bajo el sistema de autenticación, será Java1.7, empleando para la capa de presentación una GUI realizada mediante Java Swing y bibliotecas JGoodies que proporcionan una apariencia de aplicación mejorada.

Se emplearán características de java Reflection para proporcionar servicios de introspección y dotar al sistema de escalabilidad futura, siendo así fácilmente incorporables nuevas librerías de análisis de trazos, sin que ello implique volver a codificar la aplicación.

Para garantizar la seguridad del sistema, se empleará el proveedor para la Java Cryptography Extension, Bouncy Castle Crypto API (14) y se hará uso de la arquitectura PKI como solución.

La selección de la API de criptografía, ha sido debida que se trata de una API ampliamente difundida y es código libre, así como a que su estabilidad y rendimiento están sobradamente avalados.

El sistema de logs estará basado en Apache Log4j 2 (15), la nueva versión del ampliamente extendido Log4j que ha sido recientemente liberada. Esta versión cuenta con mejoras en su rendimiento, así como ciertas nuevas características como por ejemplo que la configuración se relee automáticamente sin necesidad de parar la aplicación, lo que permite cambiar el nivel de log de forma dinámica.

Es un sistema de log muy estable y de gran rendimiento que nos permite versatilidad a la hora de configurar las trazas así como su ligereza no penaliza los tiempos de respuesta durante la ejecución de la aplicación.

2.3. Análisis Funcional

2.3.1. Modelo de Casos de Uso

2.3.1.1. Catalogo Actores

Se definirán únicamente 2 roles de aplicación, el correspondiente al Usuario y el Administrador.

Ilustración 2: Rol Usuario

El rol de Usuario tendrá permisos para Registrar y Cambiar trazos nuevos, Validarse en el sistema y efectuar Logout. Será el usuario final del sistema y la interfaz con la que contará será lo más accesible posible.

Ilustración 3: Rol Administrador

El rol de Administrador tendrá permisos para Generar las claves de Administrador y las claves de la PKI y realizar la configuración del sistema en cuanto a dispositivo de entrada, validez de token de Autenticación y configuración de las Tolerancias a diferencias respecto al original, al validar un trazo.

2.3.1.2. Diagrama de Casos de Uso

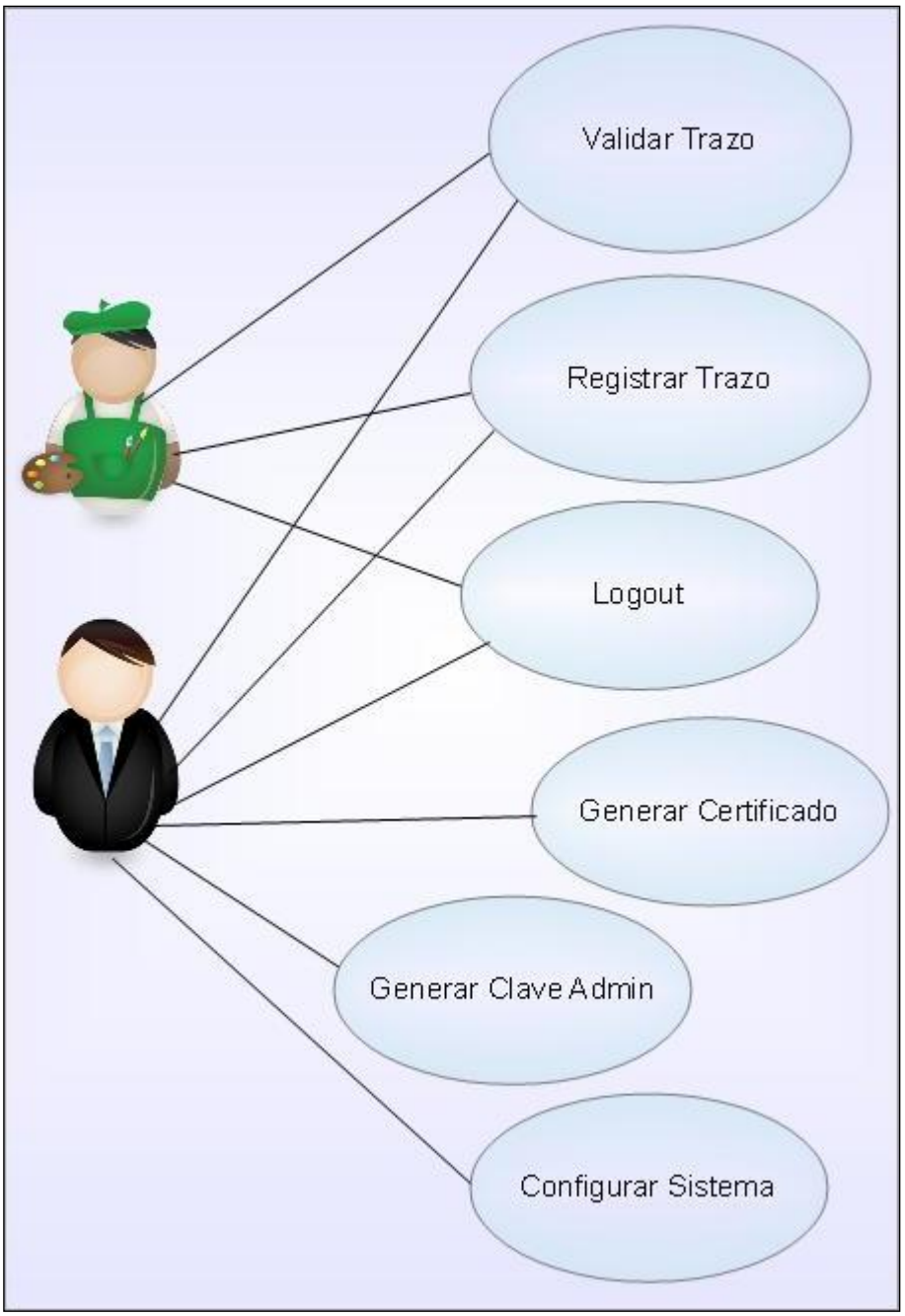


Ilustración 4: Casos de Uso

2.3.2. Prototipo de Pantallas

A continuación se muestran las principales pantallas con las que contará el sistema.

Ilustración 5: Pantalla principal

Esta será la pantalla principal que se mostrará a los usuarios al iniciar la aplicación. En esta pantalla se mostrarán mensajes *popUp* de información relevante para el usuario, así como distinta iconografía validando los procesos e indicando si han finalizado de forma correcta o incorrecta.

La pantalla de configuración del sistema se mostrará a los usuarios con permisos de Administrador. En ella se pueden encontrar los parámetros básicos de configuración con los que cuenta el sistema.

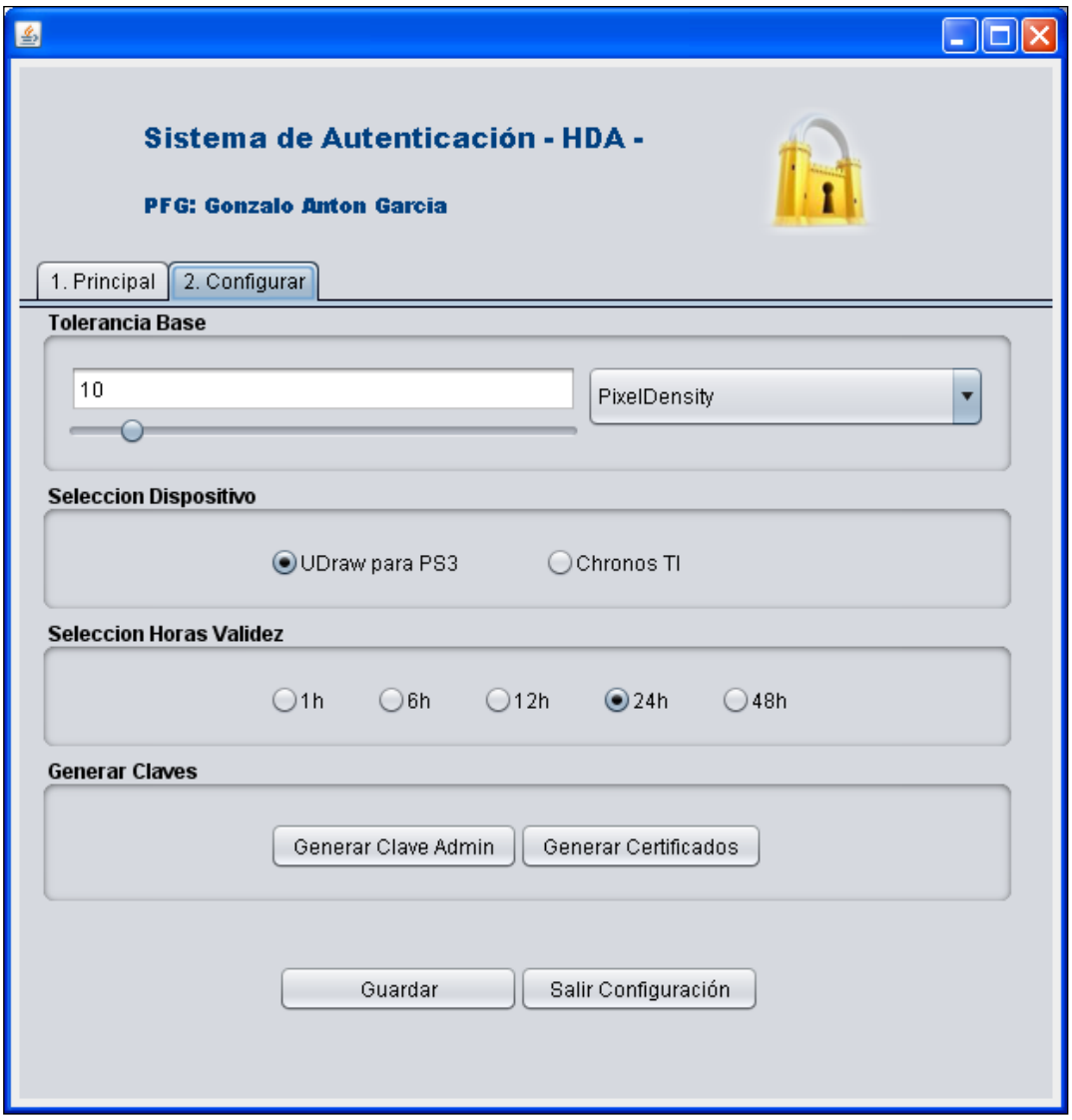


Ilustración 6: Pantalla Configuración

Esta pantalla sólo se mostrará una vez se haya proporcionado la clave de Administrador que autorice el acceso.

2.3.3. Posibles Implicaciones

Es posible que durante las pruebas de validación, el dispositivo no responda a las necesidades de los usuarios finales, proporcionando una experiencia de usuario pobre o que no alcance un alto grado de satisfacción. Estos problemas serán achacables únicamente al dispositivo en sí, pudiendo ser resueltos si se cambia de dispositivo de entrada por uno que ofrezca mayores prestaciones como por ejemplo la Tablet IntuosPen S de Wacom con un P.V.P de 54,70€ y el kit wireless Wacom de P.V.P 32,40€, ambos de venta en su tienda online (15)

2.4. Marco científico-técnico

El modelado de la aplicación, se ha basado en el funcionamiento de uno de los mecanismos de autenticación más extendidos en el mundo, como son las cerraduras convencionales. El funcionamiento de una cerradura como mecanismo de autenticación es extremadamente simple: el sistema permanece cerrado hasta que llega un usuario. La cerradura le presenta el desafío, si el usuario posee el token correcto para superarlo, el sistema le da por autenticado y se abre.

2.4.1. Descripción de la Solución

El sistema llave-cerradura, se puede analizar y descomponer en unidades lógicas, hasta llegar a la esencia del funcionamiento de dicho sistema.

Como se puede observar en la siguiente figura, la cerradura, internamente está compuesta por un número de cilindros, de longitud variable, que deben permanecer alineados para que ésta se abra.

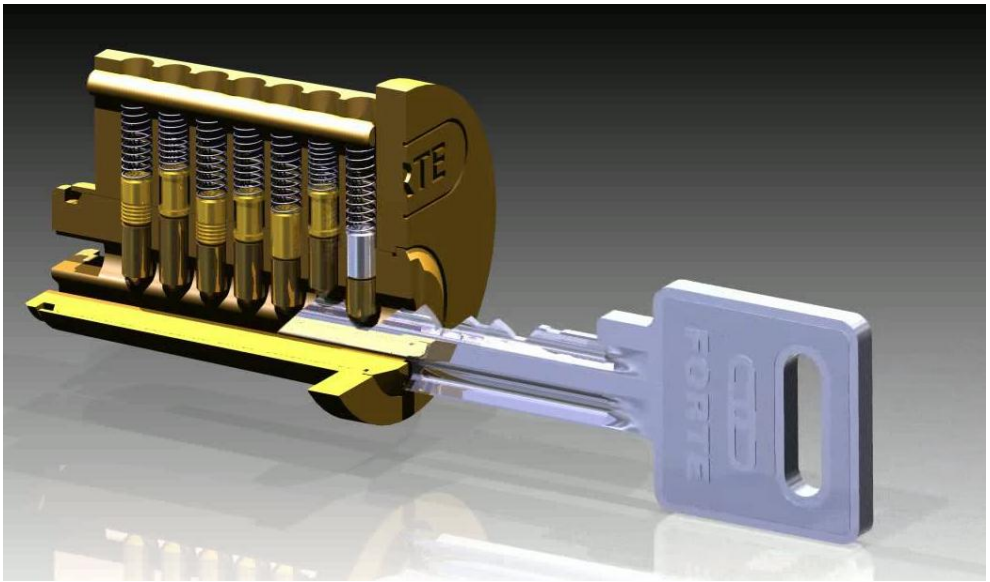


Ilustración 7: Cerradura convencional

La cerradura permanece cerrada mientras los cilindros no se encuentren alineados correctamente. Es aquí donde la llave desempeña su función. La llave dispone de unas muescas de distinta profundidad, cuyo objetivo es compensar las longitudes de los cilindros para alinear el sistema y así poder abrirlo.

Así pues el sistema llave-cerradura, podría modelarse matemáticamente como un sistema formado por dos conjuntos, el conjunto X (Cerradura) y el conjunto Y (Llave).

X está formado por elementos x_1, x_2, $x_3 \ldots x_n$

Y está formado por los elementos y_1, y_2, $y_3 \ldots y_n$

Los elementos x_1, x_2, $x_3 \ldots x_n$ representan la longitud de cada cilindro de la cerradura.

Los elementos y_1, y_2, $y_3 \ldots y_n$ representan la profundidad de cada hendidura de la llave.

Tenemos pues:

$$\begin{aligned} f: \quad X &\rightarrow Y \\ x &\rightarrow y = f(x) \end{aligned}$$

Ecuación 1: Sistema Llave-Cerradura

En concreto se puede hablar de función biyectiva, ya que se cumple:

$$\forall y \in Y \ : \quad \exists! \ x \in X \ / \quad f(x) = y$$

Ecuación 2: Función biyectiva

O dicho de otro modo, para todo valor de y perteneciente al conjunto Y, existe un único valor x perteneciente a X tal que la función f aplicada a x es igual a y.

Así mismo se puede garantizar que su función inversa es también biyectiva, por lo que nos encontramos ante una función real biyectiva.

En nuestro caso concretamente, la función que relaciona ambos conjuntos es la siguiente:

$$f(x) = -x$$

Ecuación 3: Relación Llave-Cerradura

Esto es así, ya que al aplicarla, para cada cilindro de la cerradura, debe existir una y sólo una hendidura en la llave y para cada hendidura en la llave debe existir un y sólo un cilindro en la cerradura y la función que relaciona ambos conjuntos es que uno debe ser de signo contrario al otro, es decir si el cilindro tiene 5mm la hendidura debe ser de 5mm de profundidad, o -5mm, para que finalmente la línea de corte se libere y la cerradura pueda abrirse.

Una vez caracterizado el sistema, nuestro objetivo es extrapolarlo a un lenguaje de programación, para recrear su comportamiento y así proporcionar el mismo servicio de seguridad, es decir la autenticación.

Para ello deberemos decidir cómo modelar los distintos componentes del sistema original.

El sistema dispondrá de un objeto Cerrojo, que estará formado por un conjunto de elementos o parámetros y que proporcionará al mundo exterior una funcionalidad denominada Abrir cuyo parámetro de entrada será una Llave.

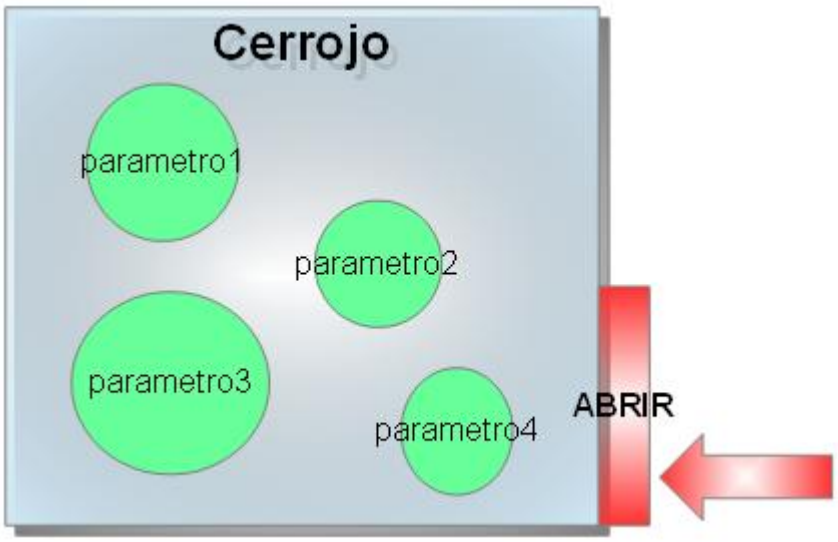


Ilustración 8: Modelo Cerrojo

El modelado de la Llave es más simple, ya que se puede representar como un conjunto de parámetros sin más.

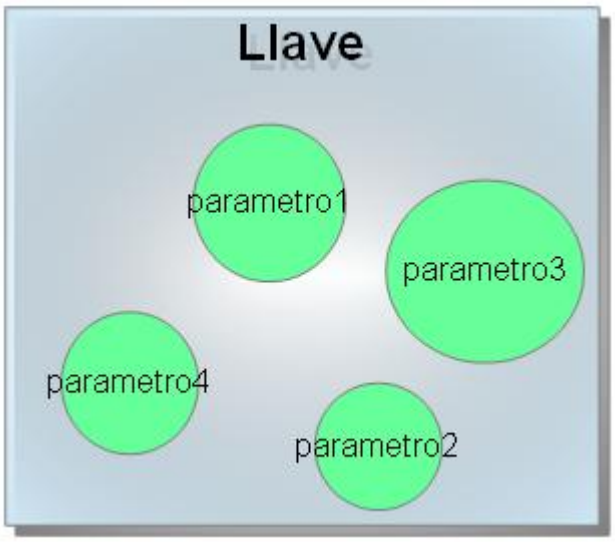


Ilustración 9: Modelo Llave

Los parámetros que componen cada uno de los sistemas deben estar, según nuestro modelo matemático, relacionados mediante una función $f(x)$. En nuestro caso y para simplificar al máximo, usaremos la función:

$$f(x) = x$$

De esta manera se establece una relación de equidad entre los elementos del subsistema Cerrojo y sus homólogos en el subsistema Llave.

El proceso de autenticación será satisfactorio, cuando el resultado de aplicar la función que relaciona el subsistema Llave y el subsistema Cerrojo sobre todos los elementos se verifique. Dado que en este caso la función es f(x) = x, la autenticación será correcta cuando todos los elementos de Llave sean iguales a sus homólogos en Cerrojo.

Como resultado del proceso de autenticación, obtendremos un producto que denominaremos *Token de Autenticación*, que será consumido por terceras partes, garantía de que el poseedor de dicho token ha superado una prueba de autenticación.

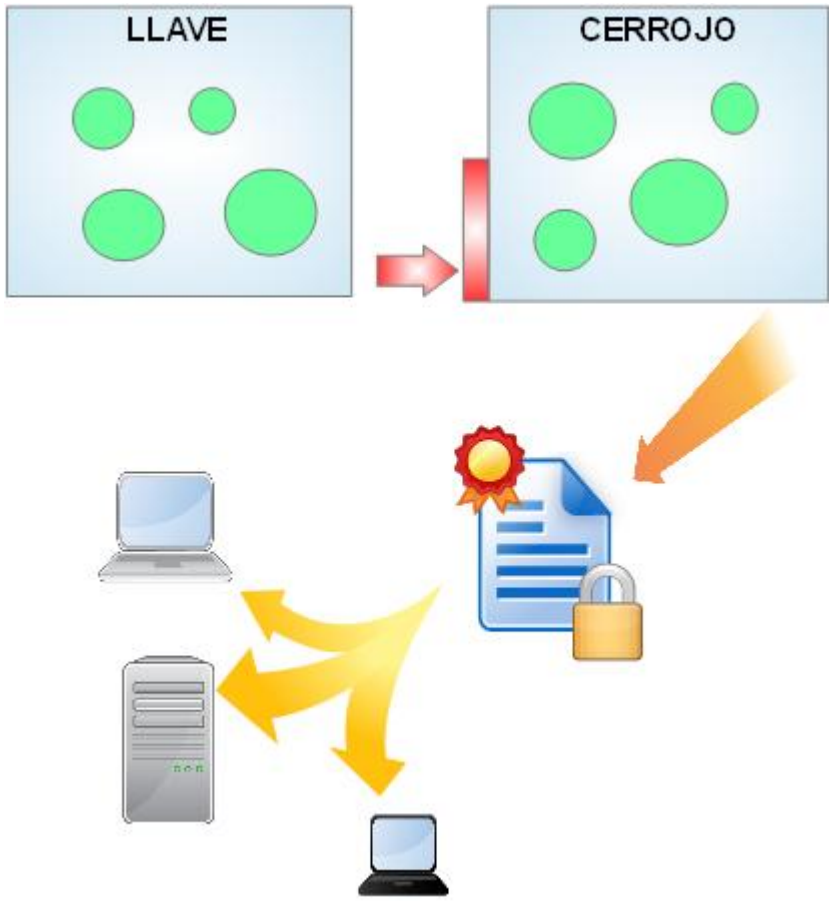


Ilustración 10: Modelo Sistema Llave-Cerradura y token

En todo este proceso, existen unos elementos indispensables y de vital importancia que son los parámetros de los que está formado cada subsistema. En el caso del sistema Llave-Cerrojo original, esos parámetros consistían en un cilindro colocado en un determinado lugar y con una longitud determinada, en nuestro caso, los parámetros van a consistir en un valor concreto para una característica medible del trazo.

Cada parámetro es un escalar, resultado de aplicar distintos algoritmos sobre el trazo realizado por el usuario. El sistema ha sido desarrollado aplicando técnicas de introspección, de tal manera que la aplicación es completamente escalable, permitiendo a futuros desarrolladores implementar sus propios algoritmos que analicen las características de los trazos que ellos consideren oportunas.

Para desarrollar sus propias bibliotecas de análisis de trazos, el desarrollador sólo deberá implementar sus clases cumpliendo con la interfaz *iKeyParameter* como se detallará en el apartado *2.4.5. Elementos Configurables* en este mismo documento.

El proceso mediante el cual se genera cada parámetro, consiste en tomar como entrada el trazo que realiza el usuario, bien sea con el dispositivo uDraw, Chronos eZ430 o cualquier otro dispositivo digitalizador, realizar un análisis cuantificando alguna propiedad de éste con un algoritmo específico y finalmente devolver un valor escalar que lo represente.

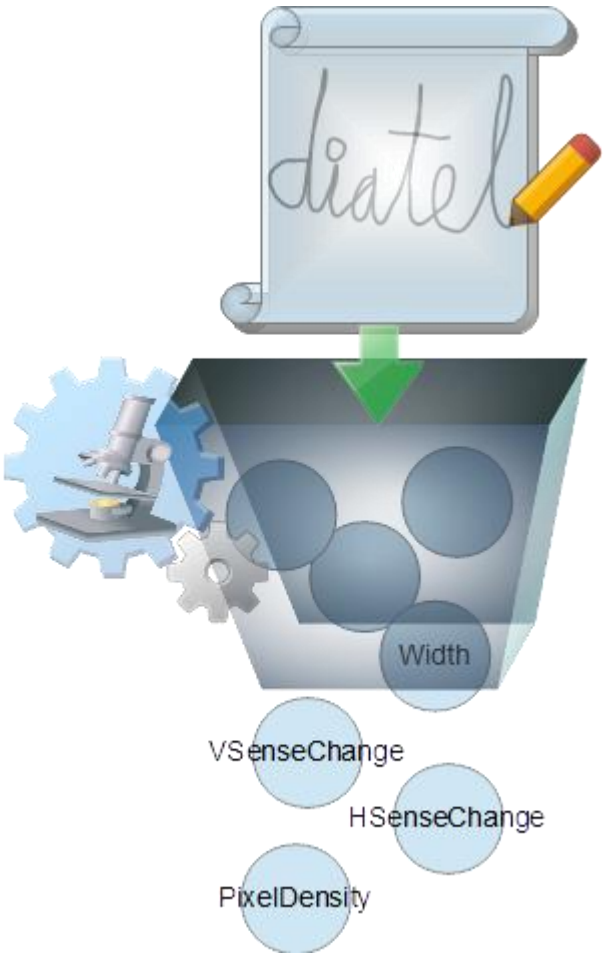


Ilustración 11: Análisis trazo

El proceso de determinar qué parámetros son los más representativos de un trazo manuscrito, es complejo y largo. Existen peritos caligráficos que estudian firmas y trazos para cotejar si dos textos manuscritos han sido realizados por la misma persona. Dichos estudios periciales, son admitidos como pruebas en juicios y poseen entidad en sí mismos.

Los parámetros que caracterizan una firma, se pueden dividir en dos grandes bloques.

- Patrones Dinámicos, que analizan características propias de la evolución del trazado según se va realizando. Son ejemplos la velocidad, los cambios de sentido, el sentido de los giros en trazos cerrados, etc...
- Patrones Estáticos, algunos ejemplos son el ancho y la altura de ciertas letras, los gestos tipo y la forma de ciertas letras.

En nuestro caso, se ha implementado al menos una muestra de cada gran grupo de patrones. Se han desarrollado distintos algoritmos, unos de carácter dinámico y otros de carácter estático, que proporcionan suficiente información para dotar de seguridad a un sistema de estas características.

Los parámetros que se han tenido en cuenta para la elaboración de esta solución son:

Width: Calcula el máximo ancho que alcanza el trazo de un extremo a otro. Es un parámetro de naturaleza estática ya que sólo influye el resultado final.

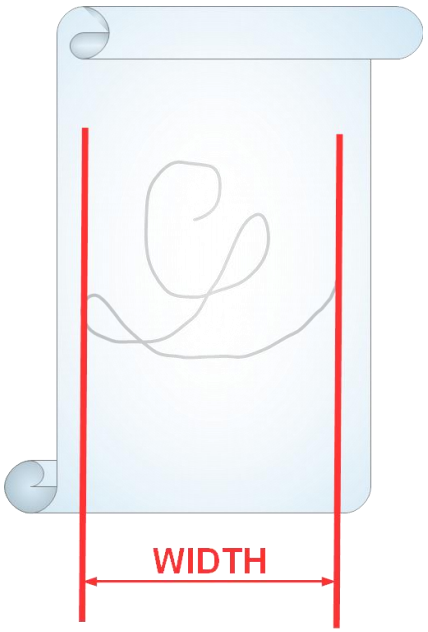


Ilustración 12: Parámetro Width

Height: Establece un escalar que representa la altura máxima que tiene el trazo realizado. Es un parámetro estático.

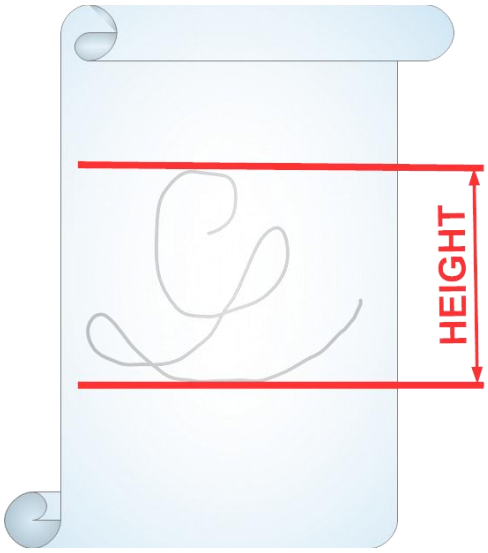


Ilustración 13: Parámetro Height

PixelDensity: Calcula la densidad de píxeles que contiene la imagen. Es un parámetro de naturaleza estática ya que en su cálculo sólo influye el resultado final del trazo.

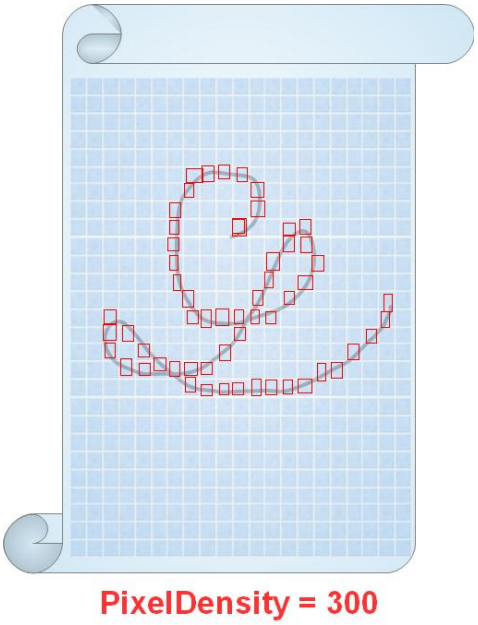


Ilustración 14: Parámetro PixelDensity

HSenseChanges: Representa el número de veces que el trazo ha cambiado de sentido horizontal durante su realización. Es un parámetro Dinámico ya que influye la secuencia de acontecimientos que ha tenido lugar durante la realización del trazo.

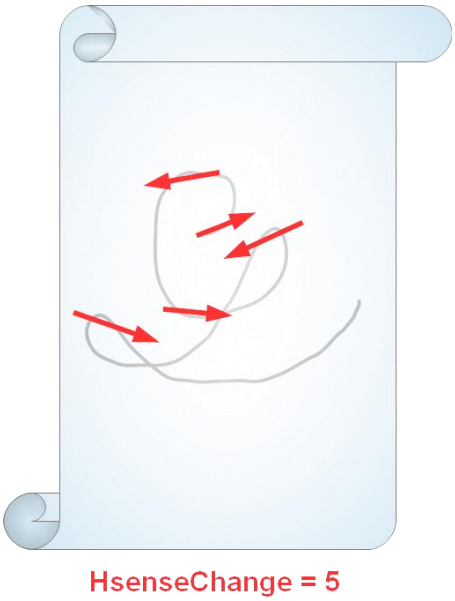


Ilustración 15: Parámetro HSenseChange

VSenseChanges: Representa el número de veces que el trazo ha cambiado de sentido vertical. Es un parámetro dinámico como en el caso anterior.

Ilustración 16: Parámetro VSenseChange

Se ha podido comprobar que con el análisis de estos parámetros, es suficiente para poder proporcionar un sistema de autenticación basado en reconocimiento de patrones biométricos, no obstante, la aplicación permite la inclusión de nuevas bibliotecas de análisis de una forma sencilla.

Durante el proceso de selección de los parámetros más significativos de un trazo, se han probado otros como analizar el centro geométrico del trazo, máxima velocidad de avance vertical detectada, máxima velocidad de avance horizontal detectada, número de trazos inconexos, presión ejercida por el usuario al escribir y la transformada discreta del coseno (DCT).

El análisis de la DCT fue desechado debido a que no aportaba grandes beneficios a nuestra solución, ya que la imagen a analizar, a parte de ser bitonal, sólo distingue 2 niveles de presión, por lo que no se apreciaba que aportase mejoras significativas al estudio del trazo. El funcionamiento de la DCT de 2-Dimensiones, es muy útil en imágenes bitonales en escala de grises, ya que al realizar su cómputo, se obtienen muchos coeficientes nulos que aportan una gran compresión de las imágenes.

La 2-D DCT-II es de la siguiente manera:

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right]\right)\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]$$
$$= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right].$$

Ecuación 4: DCT 2-Dimensiones

El proceso de cálculo de la DCT sobre una imagen, es el resultado de aplicar la DCT de 2-Dimensiones sobre bloques de 8x8 píxeles de la imagen de forma secuencial.

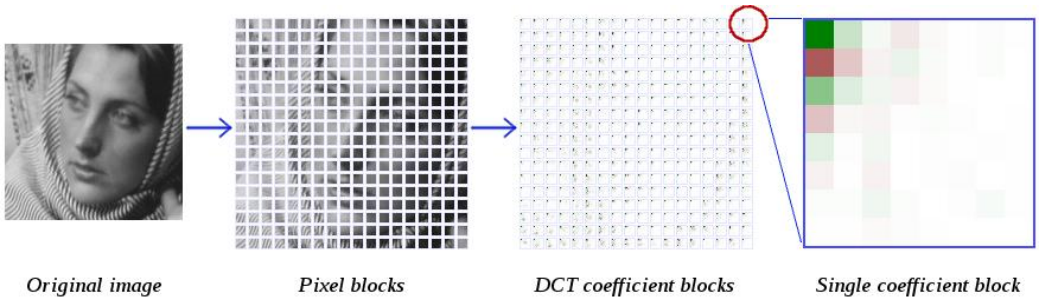


Ilustración 17: Cálculo DCT

El resultado de aplicar la DCT sobre cada bloque, es un conjunto de coeficientes de frecuencia que representan los anteriores bloques de píxeles.

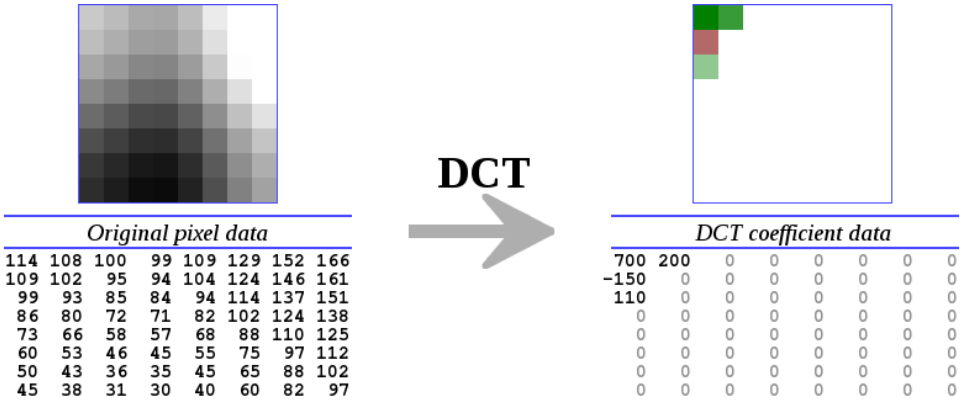


Ilustración 18: Cálculo DCT detalle

Como puede apreciarse en la figura anterior obtenida de (16), la energía se suele concentrar en frecuencias bajas, representadas en la esquina superior izquierda. Los coeficientes que representan las frecuencias más altas, se hacen 0 reduciendo así la cantidad de información presente y obteniendo una importante compresión de imágenes. Es el principio básico de algoritmos usados en la compresión de imágenes como JPG o video como MPEG.

Pero su aplicación en nuestra solución no parecía aportar grandes beneficios, ya que los dispositivos de entrada empleados, sólo contaban con 2 niveles de presión 0 y 255, con lo que no hemos apreciado mejoras sustanciales en la realización de un análisis espectral sobre el trazo, al menos mediante la DCT, que por otra parte posee un consumo de CPU bastante alto en su aplicación por software.

Capítulo 3: Diseño e Implementación

La arquitectura general del sistema se muestra en la siguiente figura:

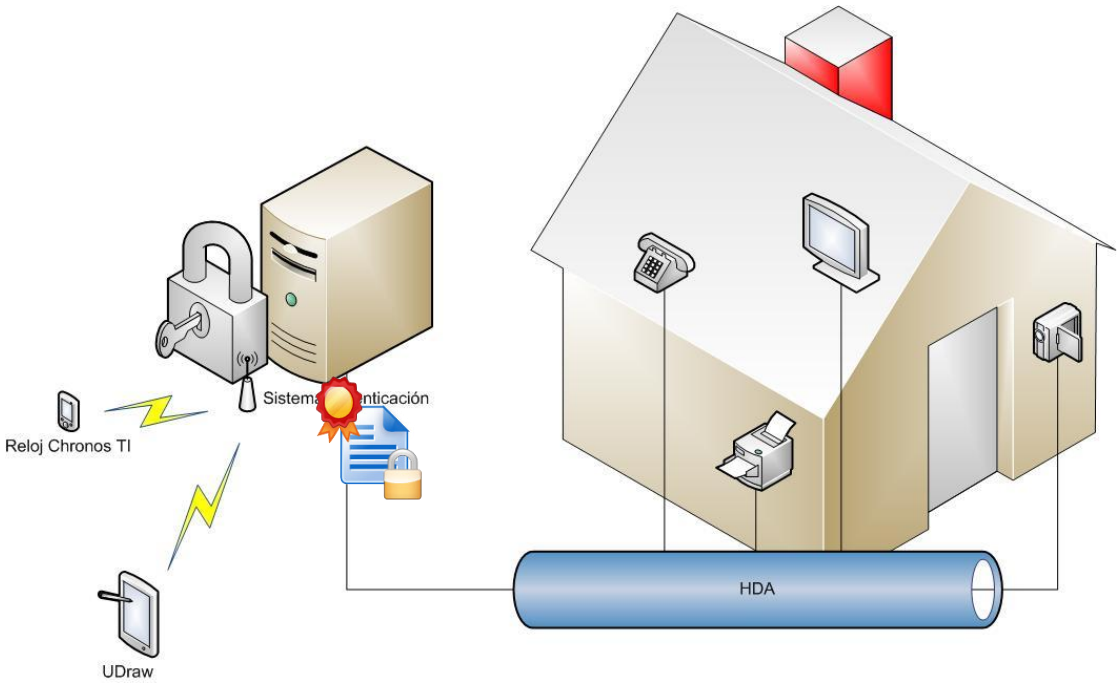


Ilustración 19: Arquitectura general sistema

Como se puede apreciar, existen dos posibles dispositivos de entrada al Sistema de Autenticación. Uno de ellos basado en la gameTablet uDraw para PS3 con su correspondiente driver para adaptarlo al uso en PC, y otro basado en el Chronos eZ430 de Texas Instruments.

El proceso comienza cuando el usuario realiza un trazo mediante alguno de ellos y éste se analiza en nuestro sistema. Una vez autenticado el trazo introducido por el usuario, se generará un fichero firmado digitalmente, al que llamaremos *Token Autenticador*, de tal manera que acredite que el usuario es quien dice ser.

El token posee una serie de datos relativos al login así como una firma que acredita que proviene del Sistema Autenticador, es entonces el HDA el que de forma activa debe comprobar que el token se encuentra dentro del periodo de validez y que la firma que contiene garantiza la integridad de sus datos. Después de estas comprobaciones, el HDA podrá dar acceso a los servicios que requieran que el usuario está autenticado.

El procedimiento de control de acceso a servicios que requieran autenticación, será pues de tipo *Pool*, siendo el HDA el encargado de acudir al repositorio, a comprobar si existe un determinado token válido para ese usuario y así otorgarle permisos de acceso.

3.1. Diseño del Modelo de Datos

La persistencia de datos en el sistema está garantizada a través del sistema de ficheros. Se ha decidido emplear ficheros en lugar de Bases de Datos, ya que la portabilidad es un objetivo a tener en cuenta en nuestra herramienta y el volumen de datos que maneja la aplicación es suficientemente bajo como para no plantearse como necesario el uso de Bases de Datos. La visión general del modelo de datos que se empleará es la de un objeto *Bean*, que posee todas las propiedades del objeto a modelar. Ese *Bean* usará unas clases denominadas Parsers cuyo fin será el de dar el formato adecuado para el volcado de datos. Para las operaciones de E/S sobre el sistema de ficheros, se ha creado unas clases denominadas DAO(Data Access Object) que encapsulan todas las funcionalidades necesarias para el tratamiento de ficheros.

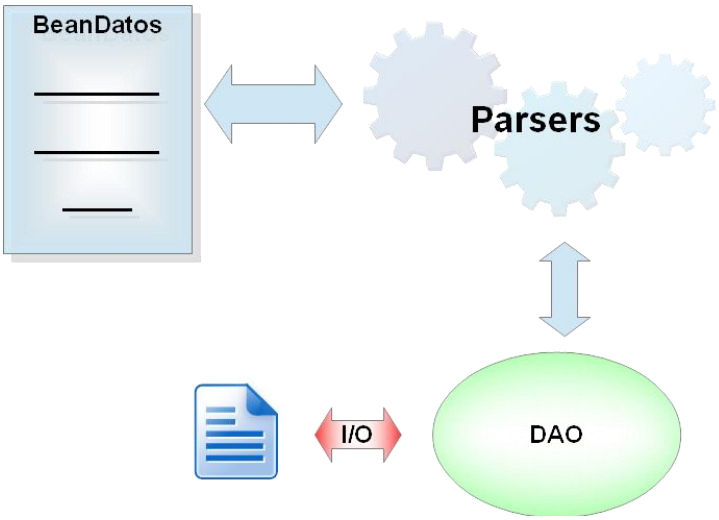


Ilustración 20: Modelo Datos

Son objetos a modelar, aquellos que contienen datos esenciales relativos a la configuración del sistema o relativos al comportamiento de este a la hora de realizar operaciones, que además deben ser persistentes incluso después del cierre de la aplicación. Los objetos modelados que poseen un *bean* que respalda su contenido son:

KeyFactoryParameterConfigBean: Encargado de encapsular los parámetros relativos a la fabricación de la llave. Especifica cómo debe crearse una llave a partir de los datos contenidos en un trazo.

LockFactoryParameterConfigBean: En él se guardan los parámetros que entran en juego a la hora de crear una Cerradura. Su estructura es similar a la de *KeyFactoryParameterConfigBean* con la salvedad de que contiene datos referentes a la tolerancia como se describirá más adelante en este documento.

ModuleConfigBean: Contiene los parámetros de configuración globales de los módulos definidos en el sistema. Se especifican los ficheros de configuración de los distintos módulos que forman la solución.

3.2. Modelo Seguridad

En los siguientes puntos se detallarán los procesos que tienen lugar para dotar de seguridad al sistema de Autenticación basado en reconocimiento de trazos.

3.2.1. Proceso de Fabricación de Cerradura

La fabricación de la cerradura tiene lugar cuando se registra un trazo en el sistema y se conserva hasta que el usuario decida cambiar el trazo registrado. Para la fabricación de las cerraduras el proceso comienza con la captura de los movimientos realizados con el periférico de entrada. En los parámetros de configuración de la *LockFactoryParameterConfigBean*, se define el número de veces que será necesario que el usuario realice el trazo. El motivo de que el usuario realice varias veces el trazo es para realizar una serie de cálculos que permitan aumentar la precisión del sistema.

El sistema analizará los parámetros de cada trazo realizado por el usuario y calculará la media. Cuantas más veces repita el usuario el mismo trazo, más fiable será luego al comprobarlo, ya que se establecerá la tendencia predominante en el trazo que desea dibujar. Junto con la media, el sistema calculará para cada parámetro del análisis, la desviación máxima respecto de la media en forma de porcentaje. A esta desviación respecto de la media le denominaremos *Tolerancia Calculada*.

Así pues, el resultado final del análisis de los trazos realizados por el usuario, será un valor medio y un porcentaje de tolerancia para cada parámetro que se haya evaluado. Quedando así establecida una banda de valores permitidos que forman la cerradura.
Este sería el resultado de 3 trazos con la letra Z, superpuestos.

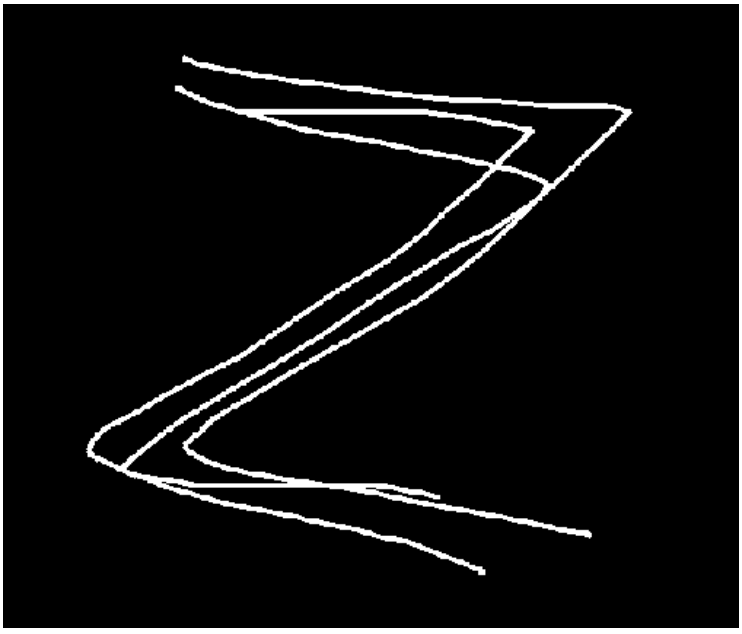


Ilustración 21: Superposición Trazado

En el capítulo de Resultados se analizarán de forma detallada algunos ejemplos como éste y se realizarán algunos comentarios. Por el momento, lo que se desea es establecer una serie de conceptos sobre el funcionamiento general del sistema.

La representación conceptual de lo que se está realizando con este tipo de análisis es la de establecer un determinado perfil con unas holguras permitidas ante ligeras variaciones de width, height, pixeldensity....

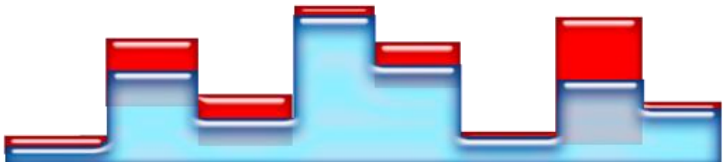


Ilustración 22: Representación Cerradura

En la figura, cada columna representa un parámetro, su altura es el valor medio de dicho parámetro, y la banda roja la tolerancia a ligeras variaciones.

3.2.2. Proceso de Fabricación de Llave

El proceso tiene lugar cuando el usuario intenta validarse en el sistema, para lo cual se le fabricará una llave en función del trazo que realice.

El proceso de fabricación de la llave, es similar al de la cerradura. Los parámetros de configuración a la hora de fabricar la llave se encuentran almacenados en la clase que modela su comportamiento *KeyFactoryParameterConfigBean*.

El análisis que se realiza sobre cada parámetro es similar, con la salvedad de que en la fabricación de la llave no existe tolerancia. La llave se fabrica con un único valor para cada parámetro que es el resultado de la media de todos los trazados.

El producto del análisis de los parámetros del trazo, nos dará un perfil que conceptualmente se representa así.

Ilustración 23: Representación Llave

Cada columna representa el valor de uno de sus parámetros, width, height, pixeldensity...

3.2.3. Proceso de Validación

El proceso de validación se produce cuando el usuario trata autenticarse en el sistema.

El sistema debe tener ya una cerradura registrada. En ese momento, el usuario realizará un trazo que a través del proceso de fabricación descrito anteriormente, se transformará

en una llave. En ese momento se comprobará que todos y cada uno de los parámetros de la llave entre dentro de la banda de valores admitidos por la cerradura.

Ilustración 24: Representación Validación

De ser así, el sistema procederá a la generación de un *Token Autenticador*, que acredite de cara a terceras partes implicadas que el usuario conoce el trazo que le autentica.

3.2.4. Proceso de Generación de token

El proceso de generación del token comienza cuando el trazo se ha validado correctamente y consiste, por una parte, en la generación de unos *Datos de Login* que contienen información relativa al usuario autenticado (la fecha desde la que es válida el certificado y la fecha hasta la que el certificado está en vigor) y, por otra parte, en la firma de esos datos para garantizar su y el no repudio de origen, que son los dos servicios de seguridad que ofrece la firma.

El proceso de firma se realiza una vez formado el cuerpo del mensaje, y consiste en calcular una función Hash, en concreto SHA256 y cifrar esta huella con la clave privada del Sistema Autenticador. La huella cifrada, se adjunta al cuerpo del mensaje y así obtenemos el token tal y como se muestra en la ilustración.

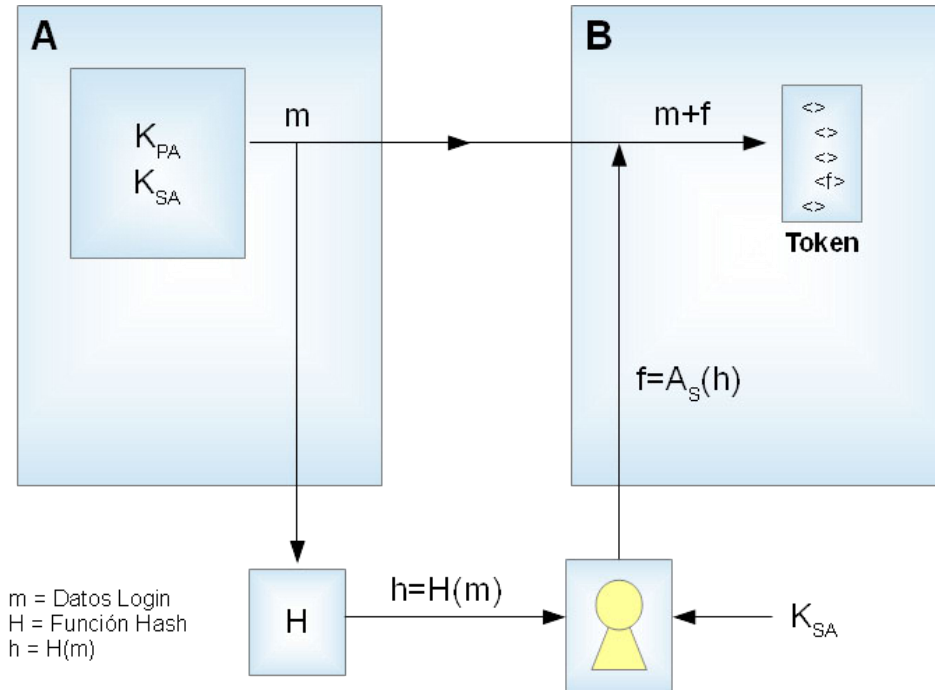


Ilustración 25: Esquema Firma

3.2.5. Flujos de Seguridad

El flujo que se sigue cuando el usuario desea Validarse en el sistema es el siguiente:

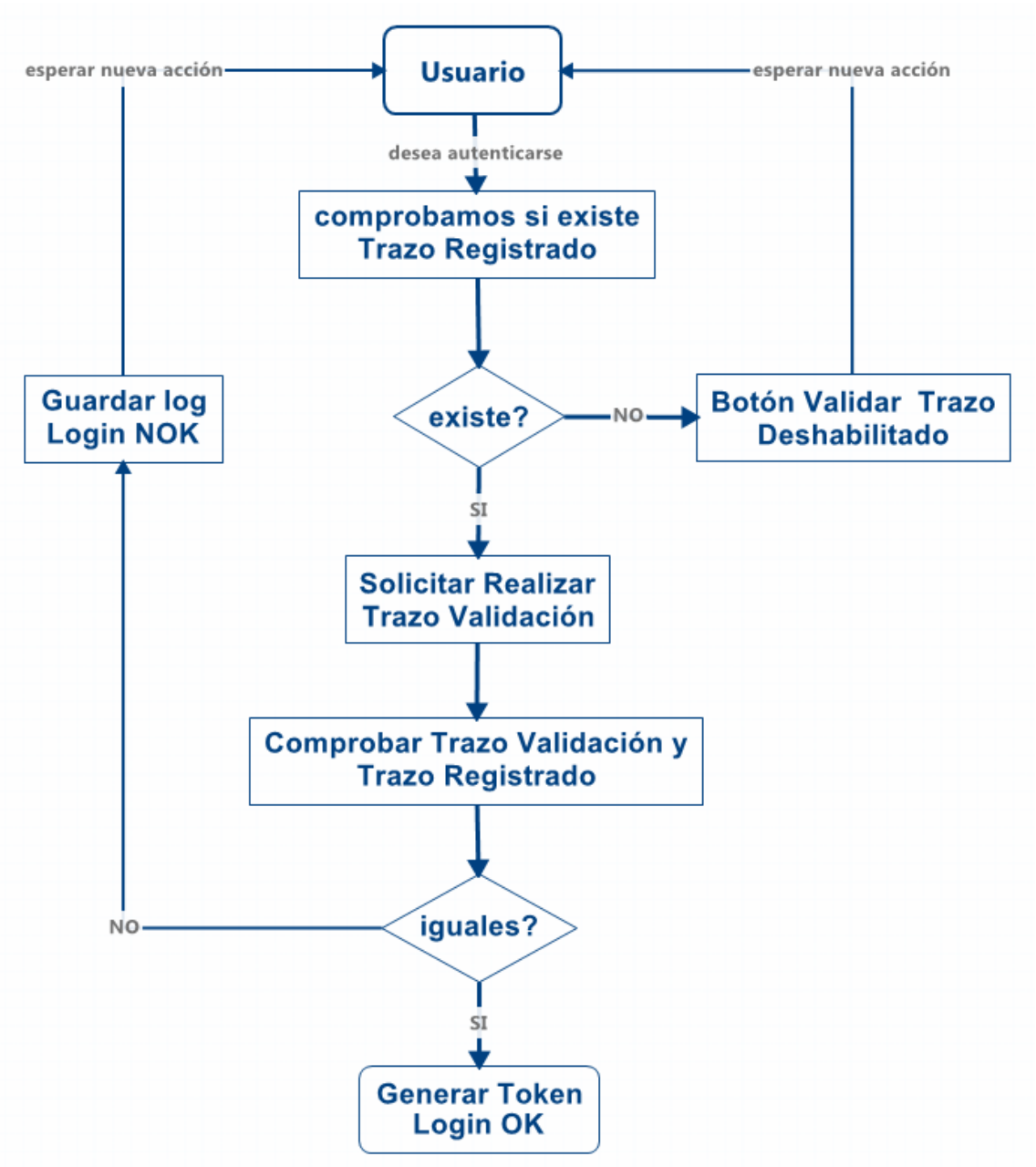


Ilustración 26: Flujo de Autenticación

El flujo cuando el usuario desea registrar un trazo en el sistema es el siguiente:

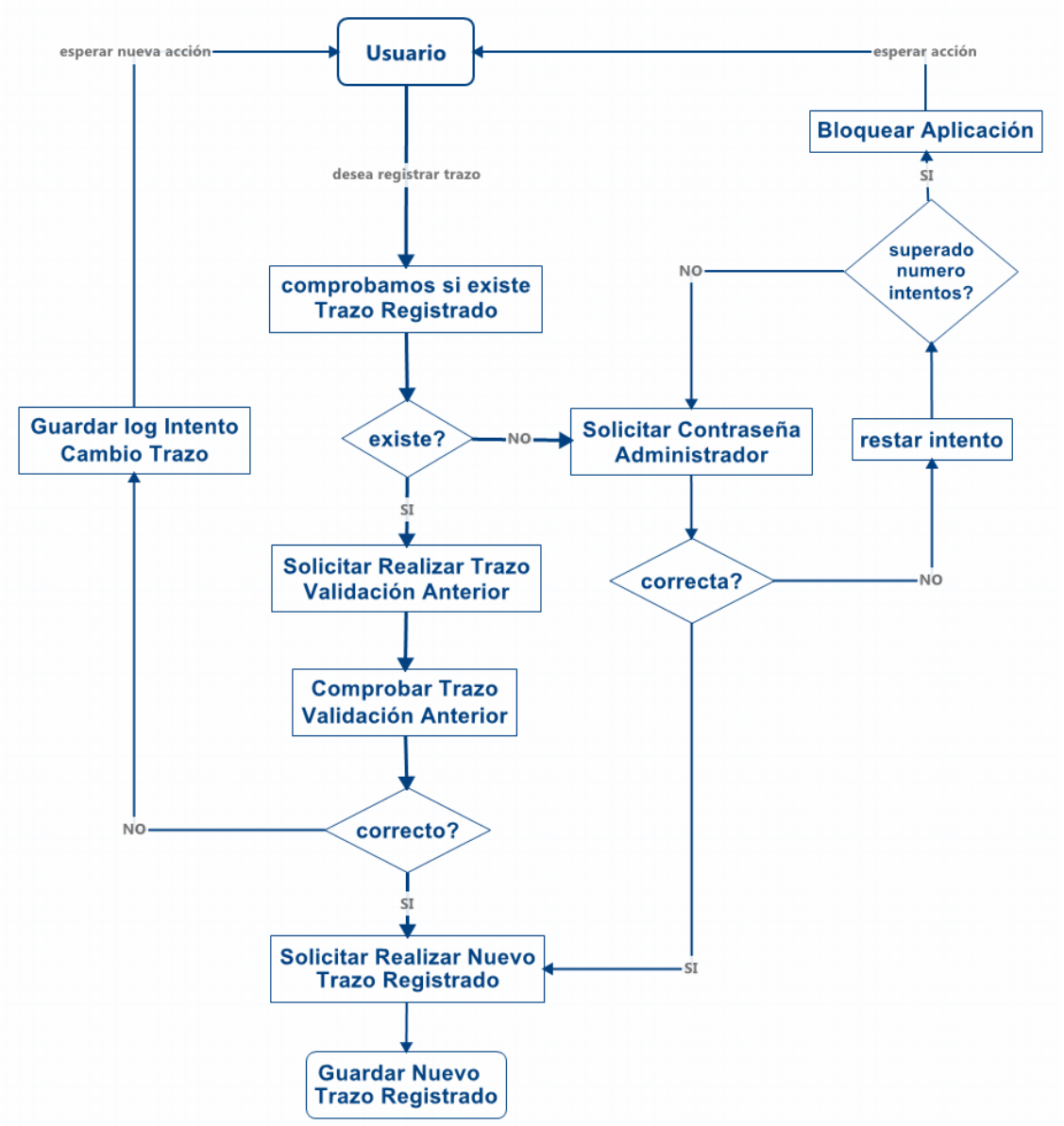


Ilustración 27: Flujo de Registro

Cuando el usuario desea configurar el sistema, el flujo a seguir es el siguiente:

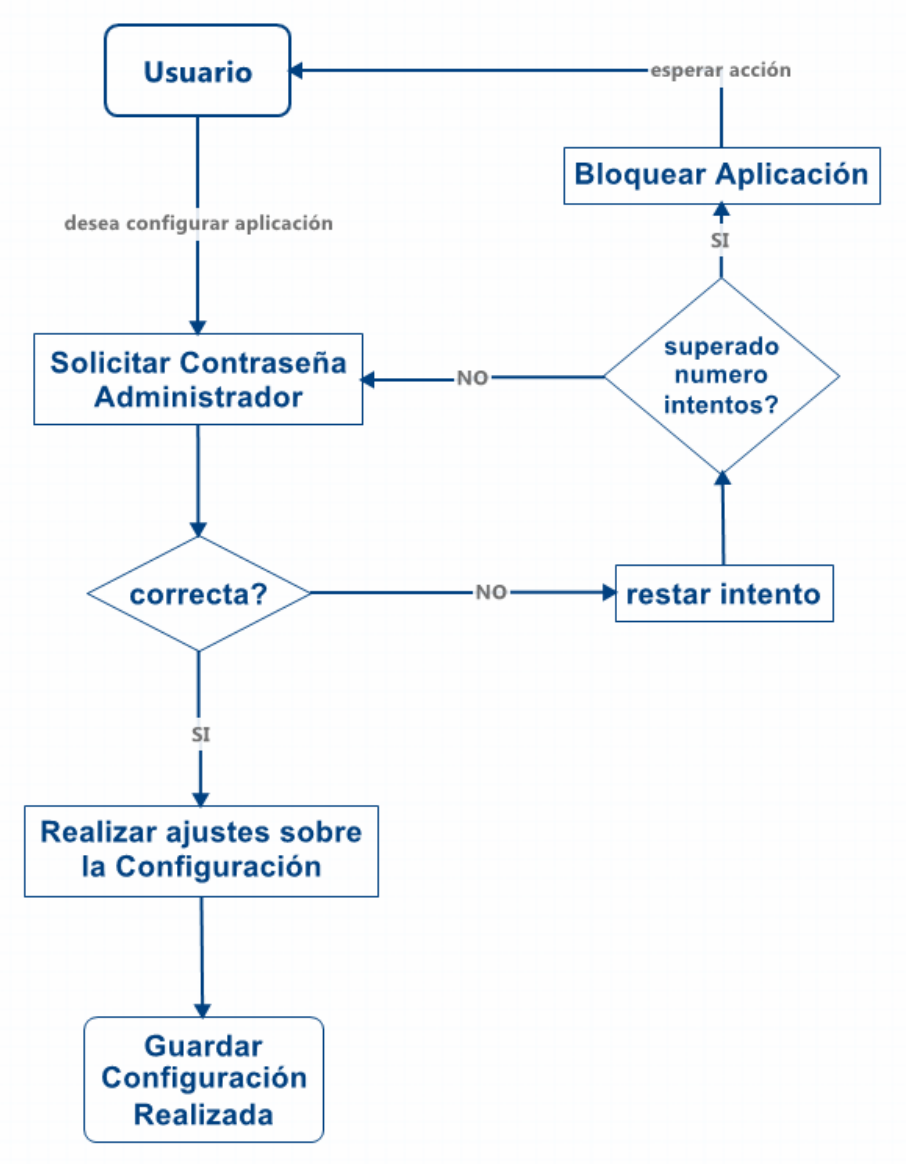


Ilustración 28: Flujo de Configuración

3.3. Modelo Clases

Dado el número de clases presentes en el sistema, y la imposibilidad de mostrar con claridad en un documento de texto todos los detalles de la estructura, se tratará de condensar las explicaciones centrándonos en los detalles más relevantes del desarrollo.

El código completo puede consultarse si fuera necesario, en los anexos que junto con esta memoria se entregarán en formato electrónico.

3.3.1. Diagramas de Clases

Por los motivos antes mencionados, mostraremos el diagrama de clases a nivel de paquete sus relaciones.

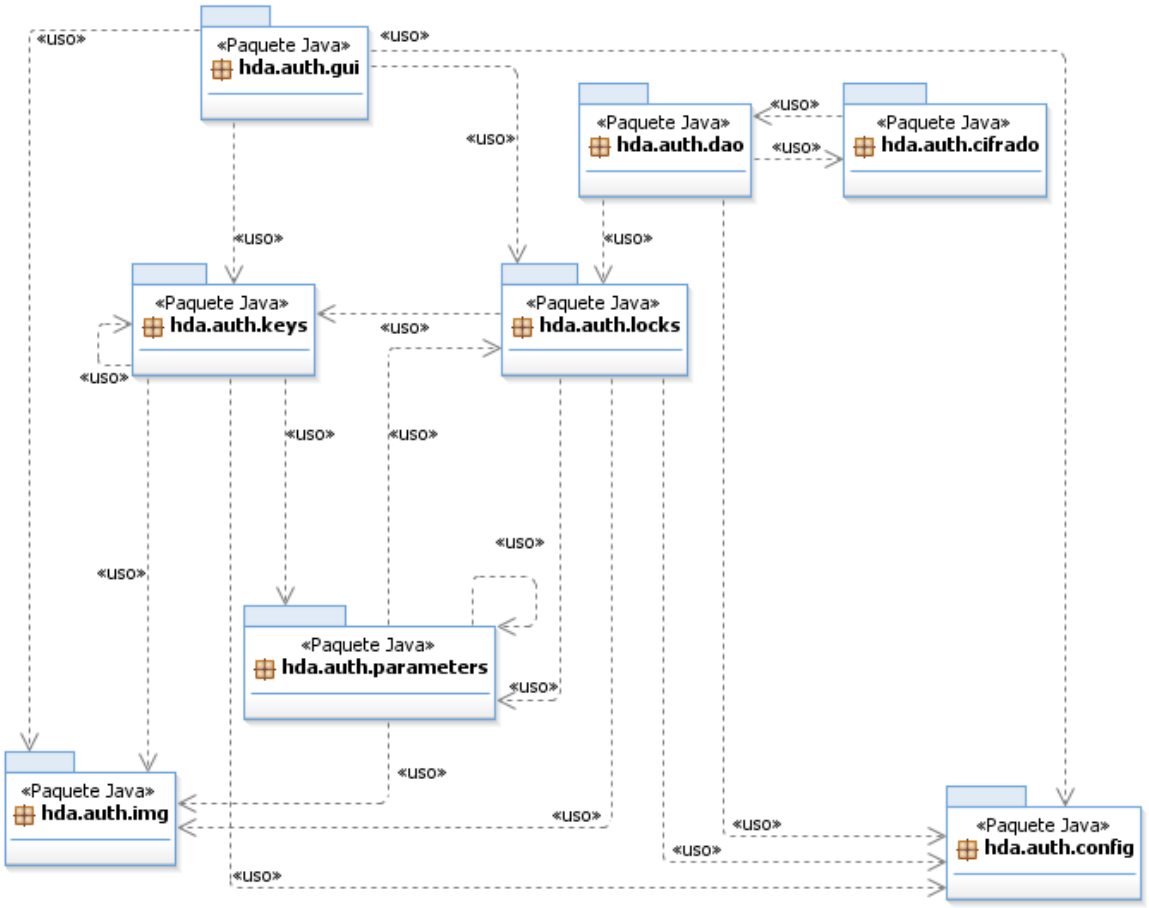


Ilustración 29: Diagrama Clases Global

A continuación pasaremos a explicar brevemente el contenido de cada paquete mostrado en la ilustración anterior.

i. Paquete hda.auth

Contiene todas las clases que forman el desarrollo. En este paquete se encuentra todo el código del proyecto del sistema de autenticación.

Se ha decidido encapsular todo el código en un único paquete global, para que su integración con el sistema de HDA ya existente, sea lo más limpia posible.

Este paquete a su vez está organizado en paquetes que gestionan distintas funcionalidades del sistema.

ii. Paquete hda.auth.cifrado

En este paquete se podrán encontrar todas las clases que gestionan la seguridad en el sistema. En su interior están las clases encargadas de la generación de claves, el cifrado simétrico y asimétrico, la obtención de identificadores únicos de máquina, así como un toolbox con las funciones más empleadas como generación de hash, etc...

iii. Paquete hda.auth.config

Este paquete contiene las clases que se emplean para gestionar los elementos de configuración del sistema.

iv. Paquete hda.auth.config.bean

Paquete contenido dentro del paquete *hda.auth.config* en el que se pueden encontrar las clases que modelan los elementos de configuración del sistema.

v. Paquete hda.auth.config.parsers

En este paquete que se encuentra en el interior de *hda.auth.config* podremos encontrar las clases encargadas de dar el formato adecuado para la serialización y deserialización de los datos. Este paquete contiene los parseadores que dan el formato adecuado a los ficheros leídos.

vi. Paquete hda.auth.dao

Este paquete contiene las clases necesarias para el acceso a los datos. Las clases en él contenidas, posibilitan la abstracción por parte de los desarrolladores de los accesos al sistemas de ficheros y encapsula funciones repetitivas y tediosas a la hora de realizar operaciones de E/S

vii. Paquete hda.auth.gui

Este paquete contiene las clases que forman la interfaz de usuario. En él se pueden encontrar las clases que forman cada elemento visual de la herramienta.

viii. Paquete hda.auth.gui.listeners

En este paquete que se encuentra dentro de *hda.auth.gui* podremos encontrar todos los capturadores de eventos presentes en el sistema. Se ha considerado necesario separar los listeners de eventos en un paquete, para facilitar las labores de mantenimiento sobre el desarrollo.

ix. Paquete hda.auth.img

Contiene las clases necesarias para el tratamiento de imágenes. Este paquete contiene las herramientas de manipulación y análisis de imágenes presentes en el proyecto.

x. Paquete hda.auth.keys

En este paquete se encuentran las clases referentes a la creación y gestión de la parte que modela la llave en el sistema Llave-Cerradura de la abstracción.

xi. Paquete hda.auth.locks

Contiene las clases relacionadas con la fabricación y la gestión de la parte referente a la cerradura en el modelo Llave-Cerradura.

xii. Paquete hda.auth.parameters

Este paquete contiene las clases que intervienen en el análisis de los parámetros del trazo realizado por el usuario. Son las clase base con las que se entrega el sistema, como ya se ha explicado anteriormente el sistema posee la capacidad de agregar nuevas bibliotecas por introspección simplemente implementando la interfaz de *iKeyParameter* presente en este paquete.

3.3.2. Definición de Clases

i. Paquete hda.auth.cifrado

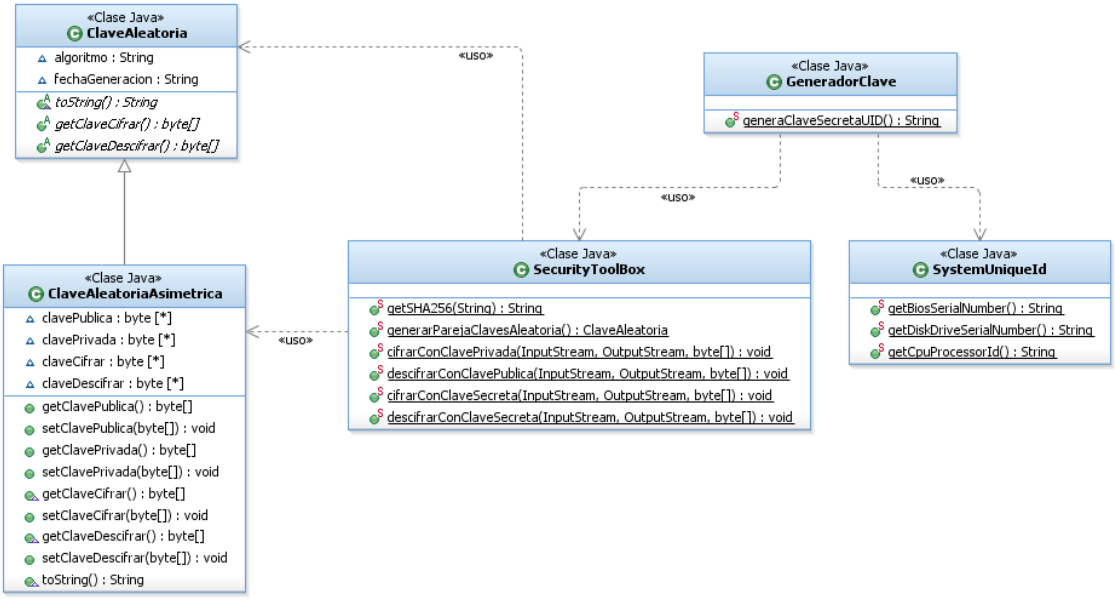


Ilustración 30: Paquete hda.auth.cifrado

Las clases contenidas en el paquete *hda.auth.cifrado*, son clases de propósito general. Cabe destacar la relación de extensión existente entre *ClaveAleatoriaAsimetrica* como una especialización de *ClaveAleatoria.*

La clase ClaveAleatoriaAsimetrica va a ser el objeto que contenga la representación de la pareja de claves usadas para dar sustento a nuestra PKI.

La clase *SecurityToolBox* contiene los métodos más empleados en referencia al cifrado que se va a emplear en la aplicación.

La clase *GeneradorClave* hace uso de los métodos contenidos en *SecurityToolBox* y a su vez se apoya en la clase *SystemUniqueId* que es la encargada de obtener los identificadores únicos de la máquina en la que se ejecute la aplicación y que serán usados como semilla en la generación de la clave secreta que se usará en el cifrado de los datos sensibles como la clave privada que se almacena en el equipo.

Se tomó la decisión de que fuese el sistema el que generase automáticamente una clave mediante un algoritmo secreto para cifrar la Clave privada ya que no era viable instar al usuario continuamente a introducir un PIN ya que el sistema requiere constante acceso a la Clave Privada para generar la firma que acompaña al *Token Autenticador*.

Así pues lo que es secreto en este caso es el algoritmo que se alimenta de los identificadores únicos de la máquina (número de serie de la BIOS, etc...) Este mecanismo es el habitualmente empleado en las licencias de aplicaciones comerciales, aunque como demuestran los KeyGen no es infalible. No obstante se ha considerado, siguiendo el principio de que los mecanismos de seguridad deben ser proporcionales al bien protegido, que en este caso era suficiente proteger la clave privada de este modo.

ii. Paquete hda.auth.config

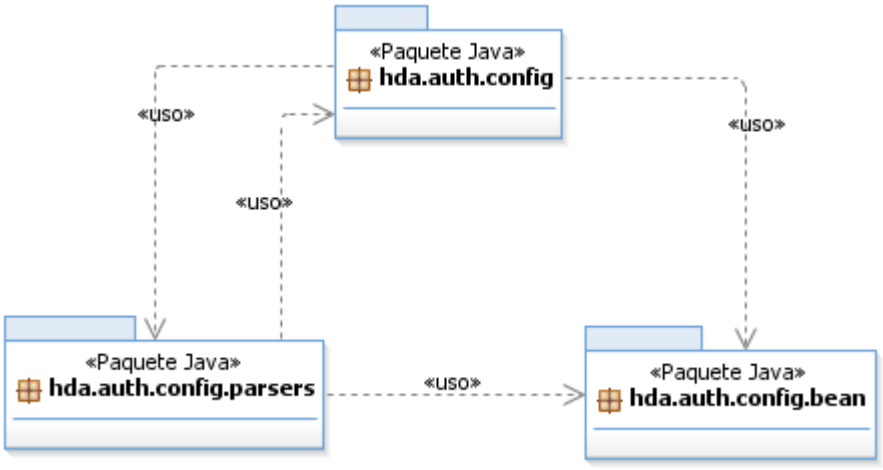


Ilustración 31: Paquete hda.auth.config estructura

El paquete *hda.auth.config*, posee a su vez dos paquetes en su interior. En este diagrama se pueden apreciar las relaciones que existen entre ellos. Las clases contenidas en el paquete raíz son las siguientes:

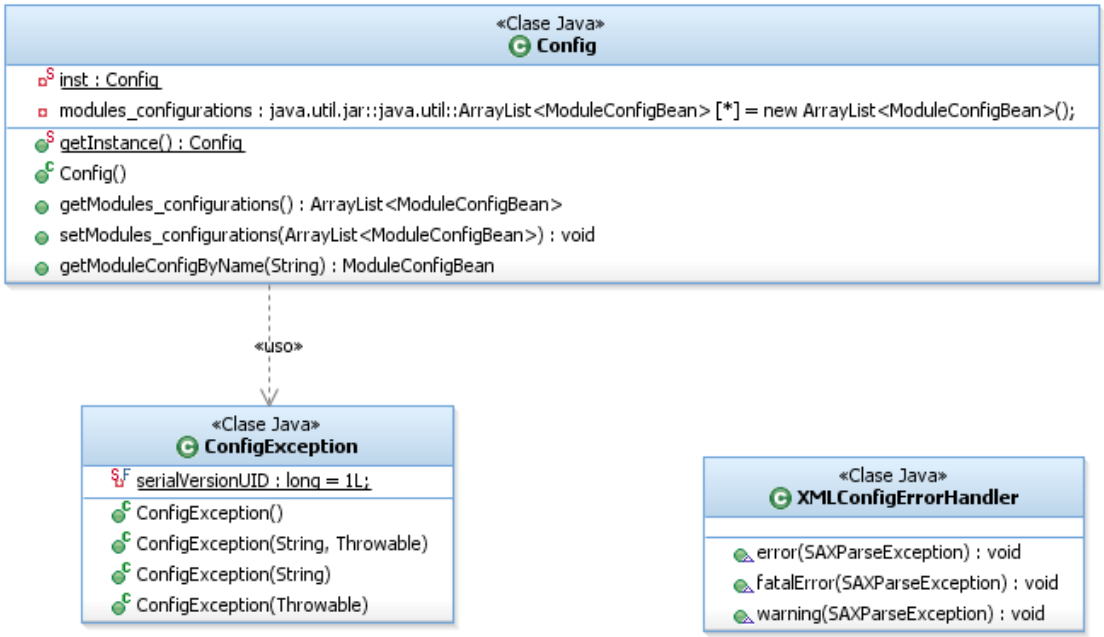


Ilustración 32: Paquete hda.auth.config

La clase *Config*, es la encargada de encapsular toda la información parametrizable del sistema y es por ello que se modela como un *Singleton* accesible desde cualquier clase del desarrollo. Cada módulo presente en el sistema guarda su configuración en un *ModuleConfigBean*, así pues el objeto *Config* guarda un *array* de *ModuleConfigBean* que pueden ser accedidos a través del método *getModuleConfigByName()*.

Posee un método llamado *getInstance()* que devolverá una referencia al objeto *Config* para poder consultar cualquier propiedad que se necesite en ese momento.

iii. Paquete hda.auth.config.bean

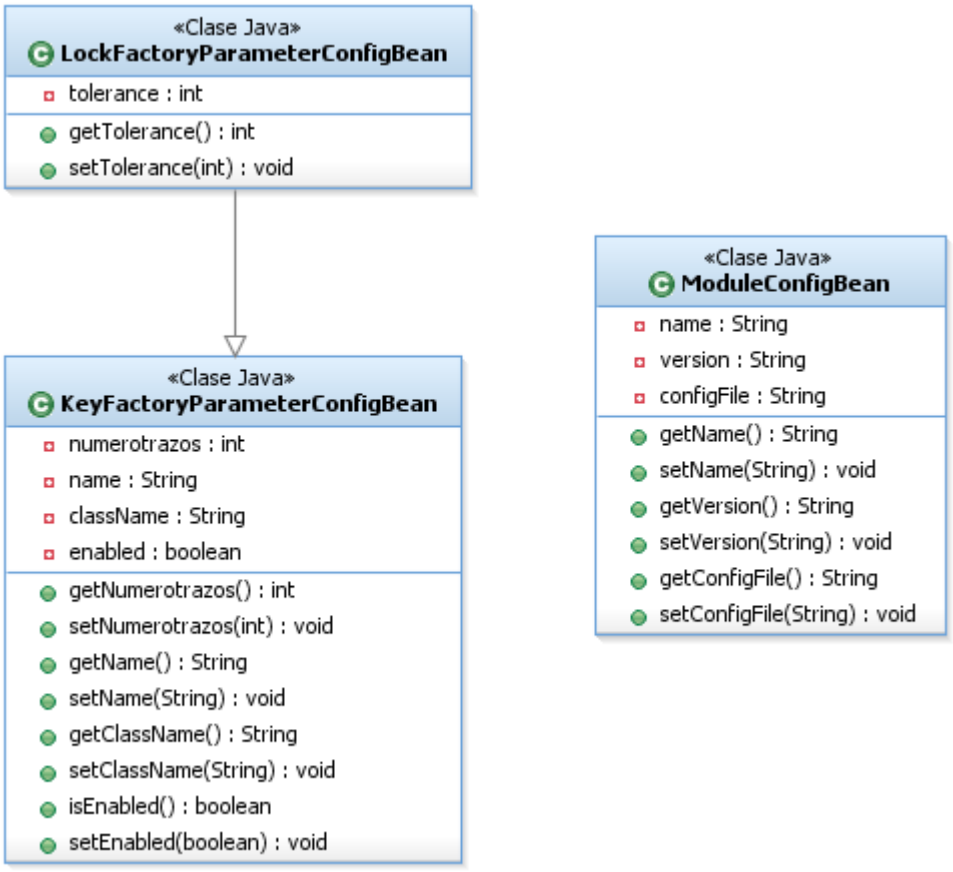


Ilustración 33: Paquete hda.auth.config.bean

Este paquete contiene el modelado de los objetos que guardan la configuración de las KeyFactory y las LockFactory, además de los módulos definidos en el sistema. Como puede apreciarse, la clase LockFactoryParameterConfigBean se ha modelado como una especialización de la clase KeyFactoryParameterConfigBean por el mero hecho de que el Lock posee una particularidad que la Key no tiene, esta es la tolerancia a ligeras variaciones como se ha explicado en anteriores puntos de este documento.

La clase *ModuleConfigBean* posee los campos que tiene cada módulo definido en el fichero de configuración MainConfig.xml de tal manera que se pueda volcar los datos y trabajar con ellos.

iv. Paquete hda.auth.config.parsers

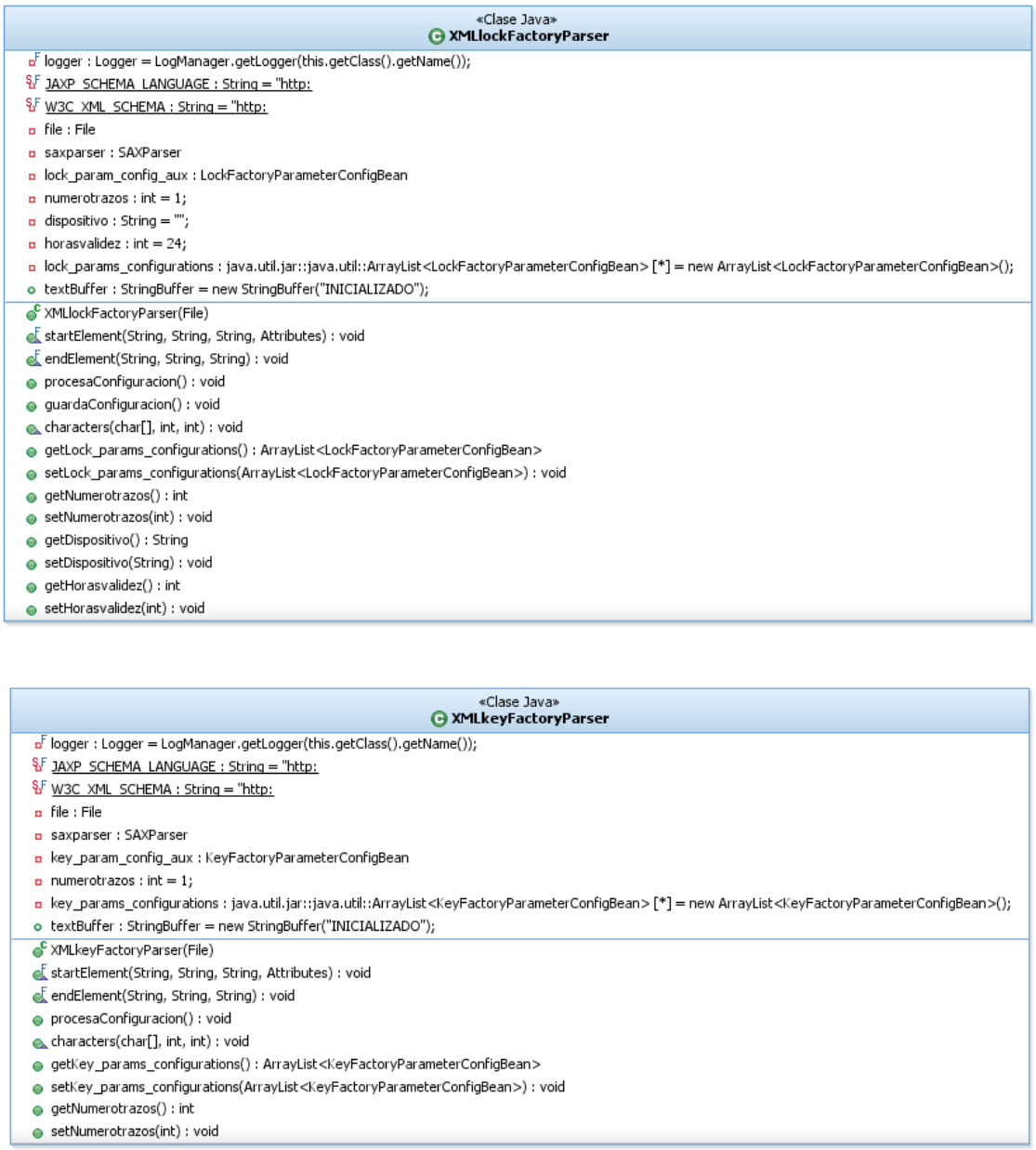
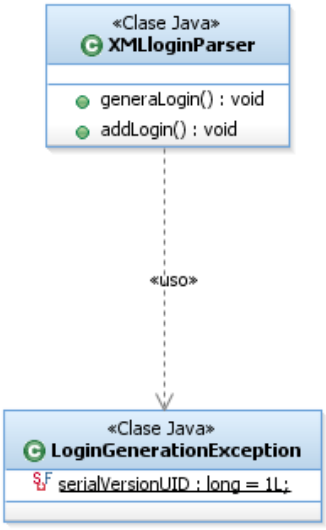


Ilustración 34: Paquete hda.auth.config.parsers(1)

En este paquete podemos encontrar las clases empleadas en el parseo de los ficheros xml a estructuras de datos java. Los parsers poseen métodos que procesan los ficheros de configuración, los validan contra estructuras xsd y finalmente los vuelcan en clases *bean* que modelan dichas estructuras.

Para la lectura de los ficheros se ha decidido emplear SAX por su comodidad. Así como para su manipulación y escritura se ha decidido emplear un modelo basado en DOM porque se ha considerado lo más idóneo.



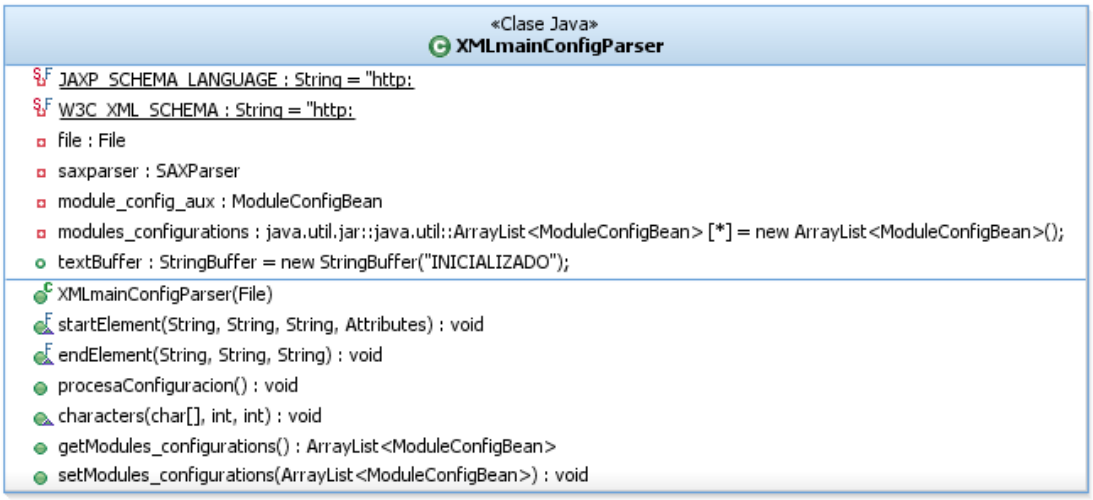


Ilustración 35: Paquete hda.auth.config.parsers(2)

Junto con las clases anteriores, se pueden encontrar en *hda.auth.config.parsers* la clase que da soporte a la estructura del Login.

v. Paquete hda.auth.dao

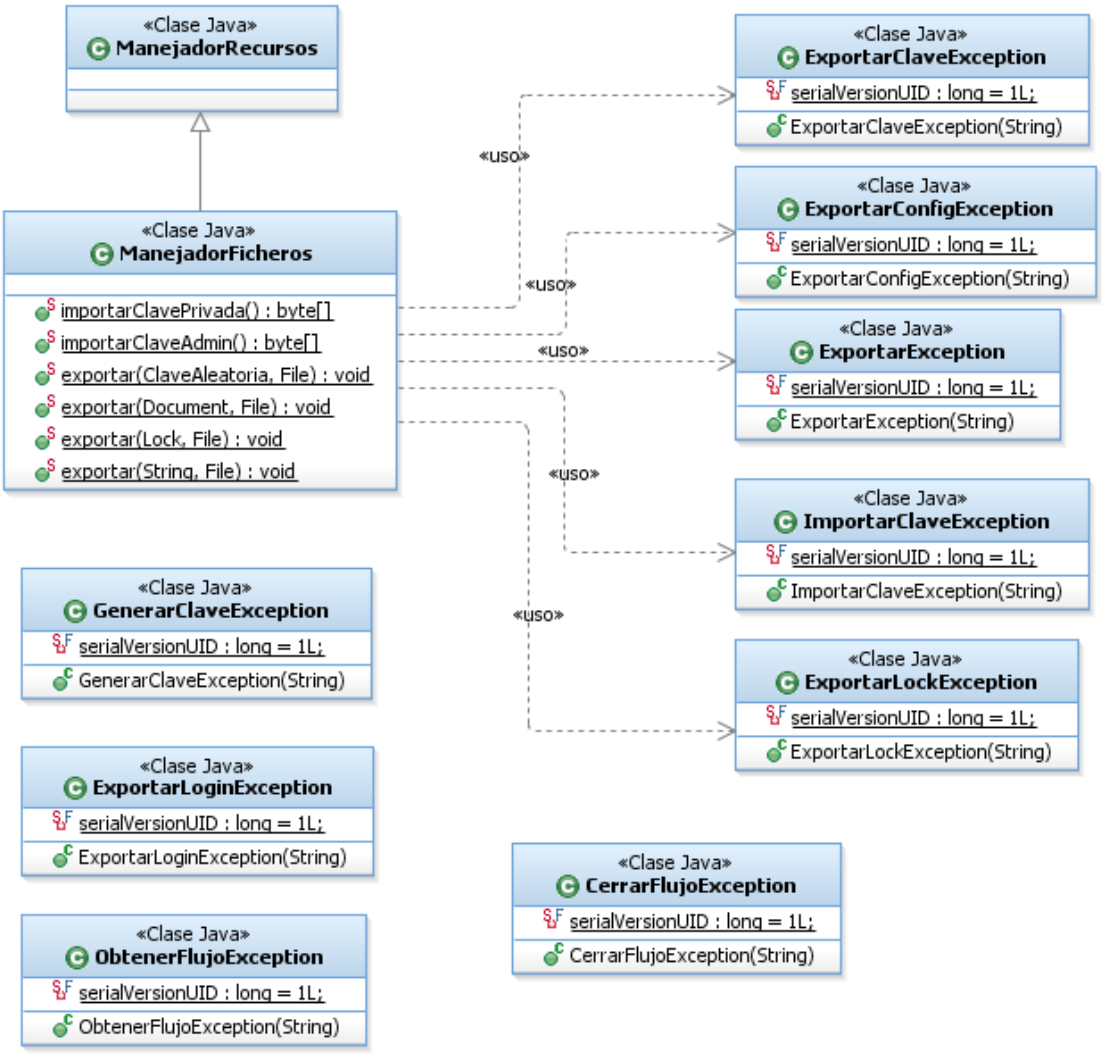


Ilustración 36: Paquete hda.auth.dao

Este paquete contiene las clases relacionadas con el Data Access Object(DAO) y como su nombre indica, encapsula todos las operativas necesarias para el acceso a datos.

Se ha querido en este caso extender de una clase genérica denominada *ManejadorRecursos*, porque aunque en este caso la persistencia de datos se realiza sobre el sistema de ficheros, pudiera ser de consideración adaptarlo a una BBDD o un servidor FTP si en futuros desarrollos se considerase necesario. Es por ello que se ha querido dejar abierta esa posibilidad en el diseño de la aplicación.

vi. Paquete hda.auth.gui

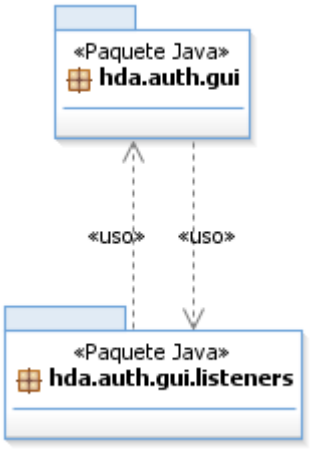


Ilustración 37: Paquete hda.auth.gui estructura

El paquete *hda.auth.gui* posee en su interior otro paquete denominado *hda.auth.gui.listener* con el que se relaciona de forma bidireccional.

Las Clases contenidas son todas las relacionadas con la interfaz de usuario y son numerosas. Se ha querido separar la parte visual de los listeners por hacer el mantenimiento de la aplicación los más legible posible y facilitar así la comprensión y modificación del código.

El sistema visualmente está estructurado como una aplicación contenida en un JFrame y sucesivos JDialog y JPanel que se van modificando, mostrando u ocultando. El sistema tiene sus iconos accesible a través del sistema de fichero, lo que lo hace fácilmente customizable.

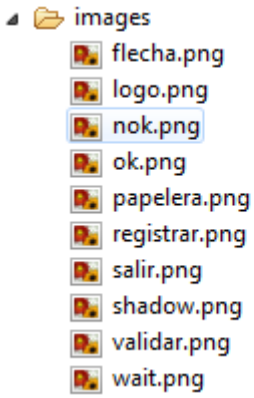


Ilustración 38: Estructura iconos aplicación

Simplemente modificando estos ficheros el usuario podrá darle un aspecto distinto a su a aplicación personalizando la apariencia.

Las clases que forman la interfaz son complejas y tienen múltiples atributos y propiedades que hacen que su aspecto sea agradable al usuario.

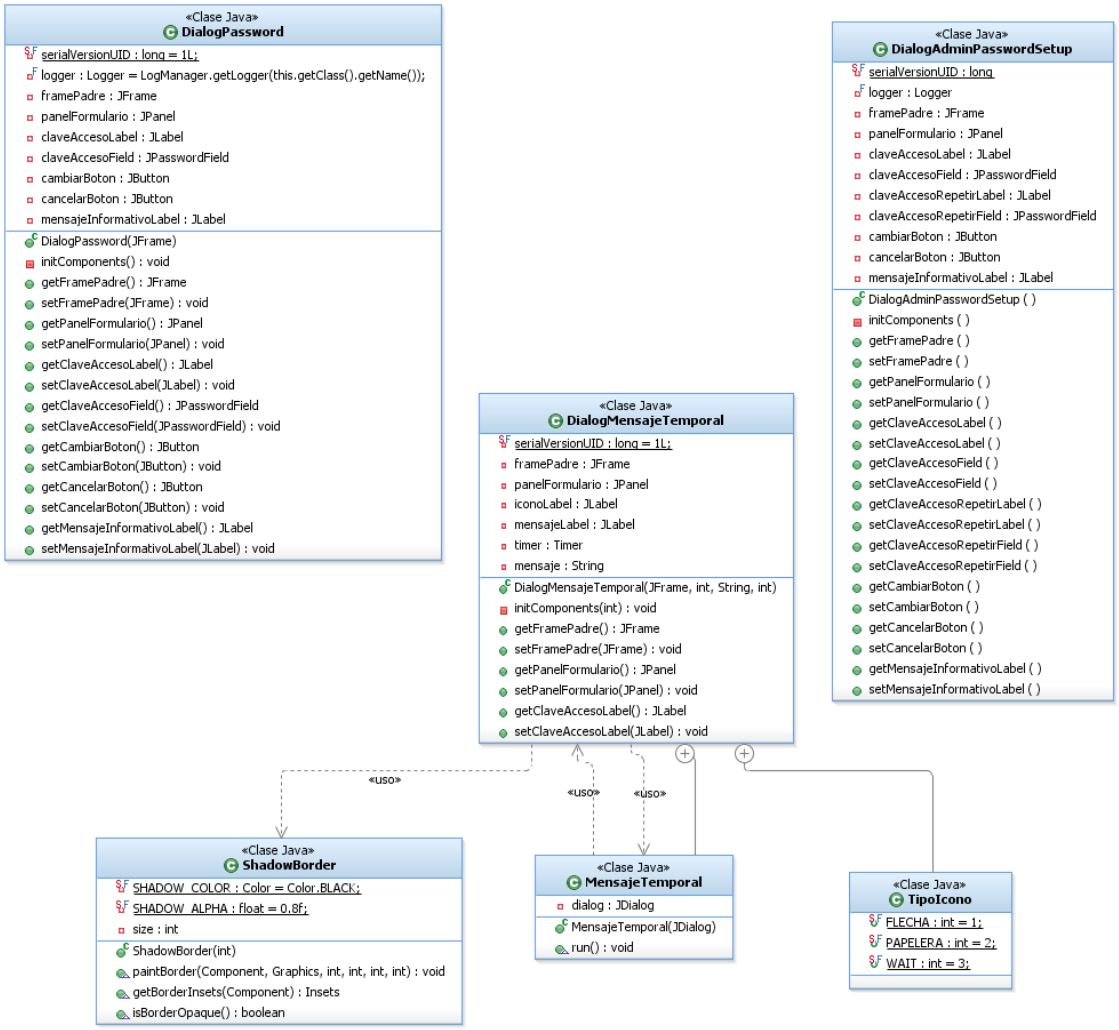


Ilustración 39: Paquete hda.auth.gui(1)

Para la creación de efectos visuales como sombras y transparencias, se han creado unos bordes personalizados implementando la interfaz del sistema *javax.swing.border.Border*.

Como detalle, hay que destacar que se ha apreciado que el efecto de transparencia, es inherente a la JVM que posea el equipo donde se ejecute, no mostrándose igual en todos los sistemas donde se ha probado la aplicación.

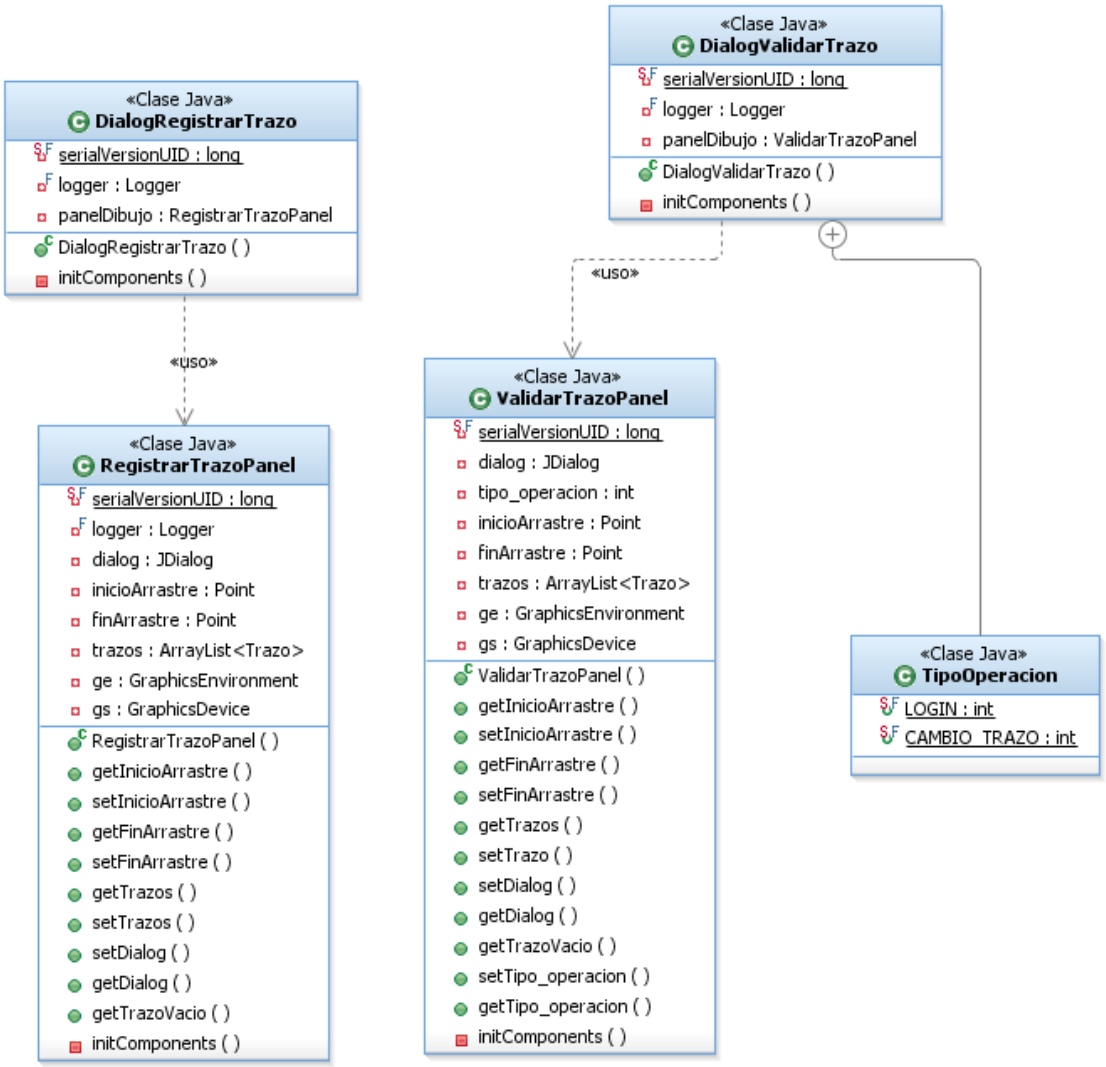


Ilustración 40: Paquete hda.auth.gui(2)

Los paneles donde se permite escribir bien para validar el trazo o para registrarlo, se ajustan a la pantalla donde se ejecute la aplicación haciendo más fácil la escritura sobre ellos.

Se tuvo al principio ciertos problemas con el foco y las ventanas que se quedaban por delante que fue solventado estableciendo una jerarquía bien definida de JDialog padre y JDialog hijos, para que al cerrar uno se volviese siempre al punto donde se había creado, evitando problemas de foco.

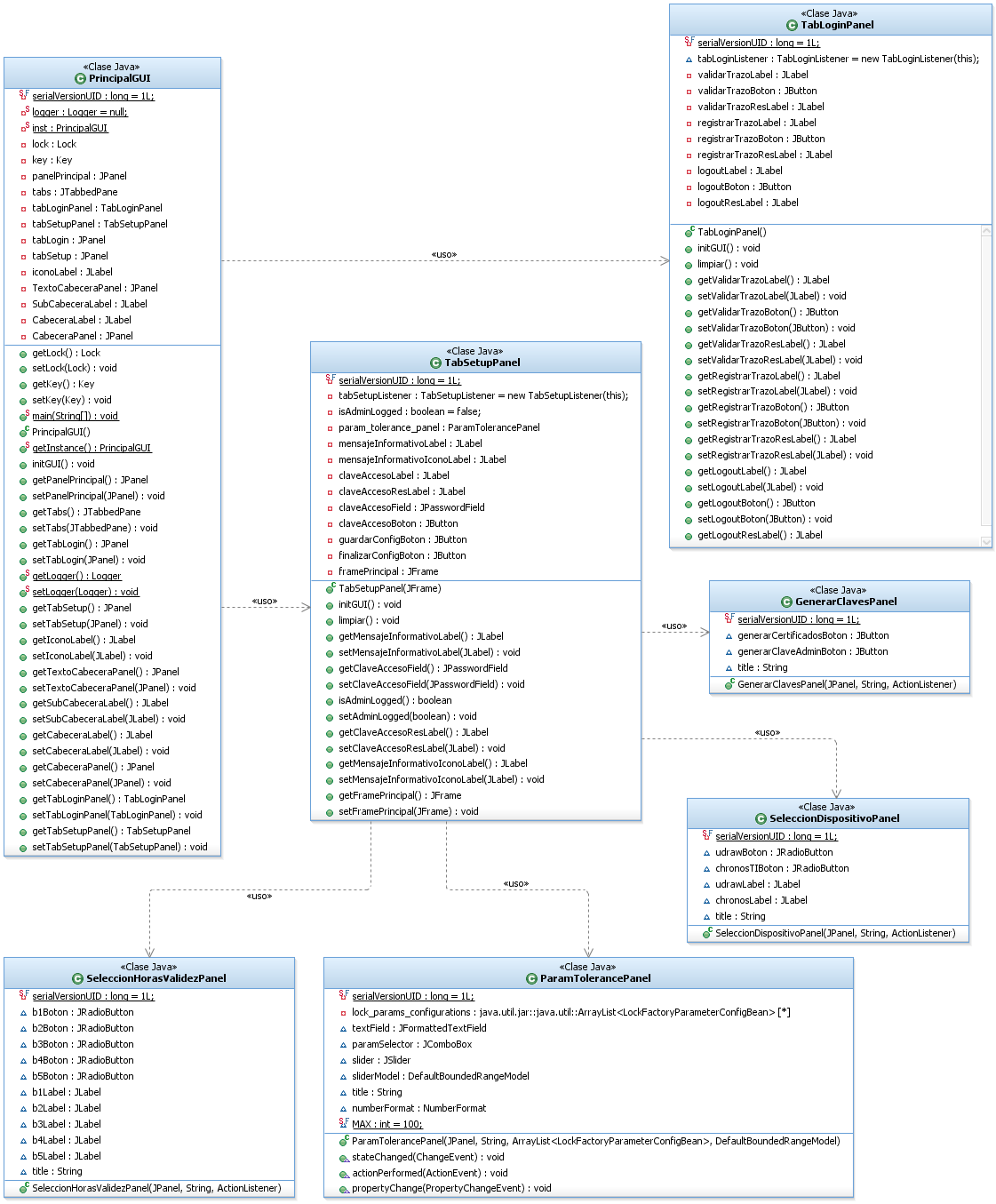


Ilustración 41: Paquete hda.auth.gui(3)

La pantalla principal está representada por la clase PrincipalGUI, el único JFrame de la herramienta. Como puede apreciarse, tiene dos paneles con pestañas donde residen las funciones de cada rol de usuario. Una pestaña para usuarios estándar y una pestaña para administradores, protegida por contraseña, donde se podrá configurar los parámetros de funcionamiento.

vii. Paquete hda.auth.gui.listener

Ilustración 42: Paquete hda.auth.gui.listener

Este paquete contiene los listeners de todos los eventos del sistema. Se han dividido clasificándolos por las interfaces en las que quedan a la escucha de eventos.

viii. Paquete hda.auth.img

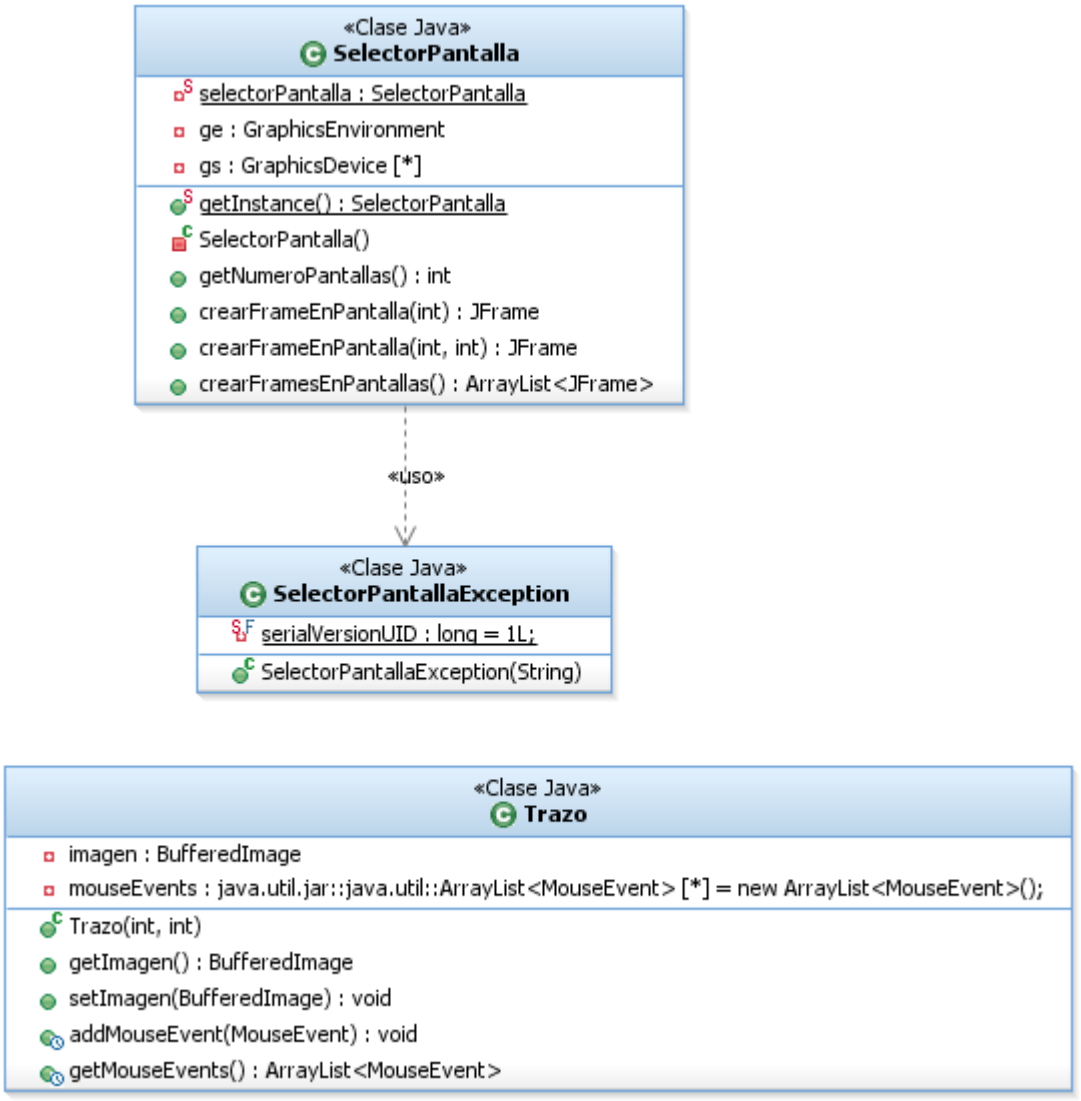


Ilustración 43: Paquete hda.auth.img

En este paquete podremos encontrar las clases relacionadas con el tratamiento de imágenes. La clase *SelectorPantalla* es la encargada de obtener las propiedades de la pantalla para adaptar la aplicación a su tamaño

La clase Trazo, contiene toda la información capturada por el sistema en referencia al trazo realizado. Tiene por así decir los datos en crudo que posteriormente serán analizados.

ix. Paquete hda.auth.keys

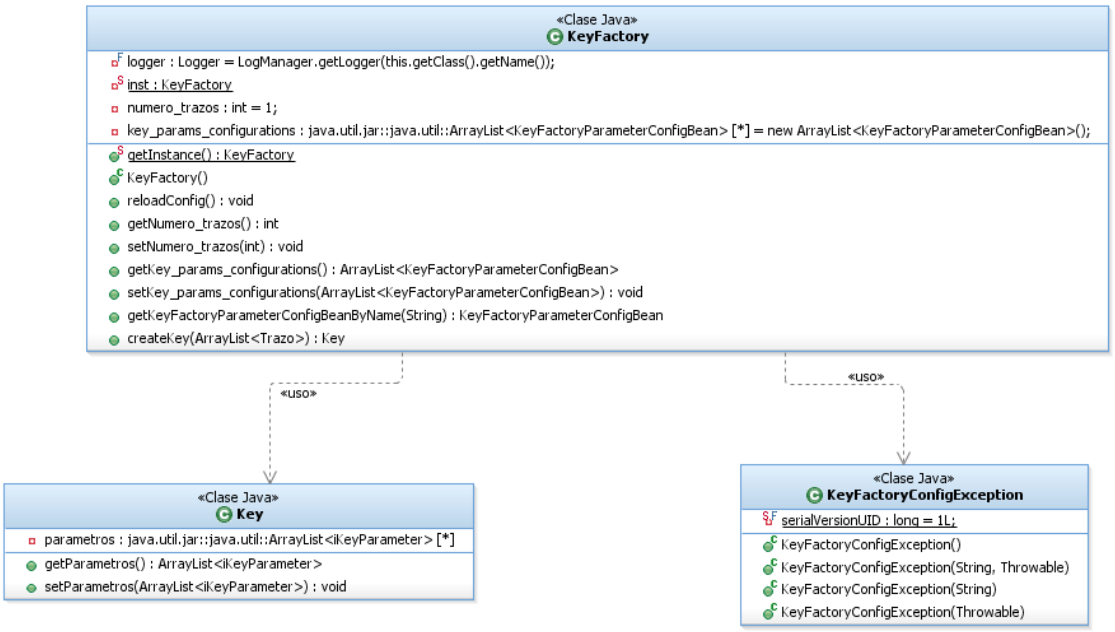


Ilustración 44: Paquete hda.auth.keys

Este paquete contiene las clases relativas a la generación y modelado de las keys. La clase *KeyFactory* es la encargada de generar las llaves a través de su método *createKey(ArrayList<Trazo>)*. La clase está implementada como un *singleton* que usa introspección y java reflected para la fabricación. La fabricación se apoya en los ficheros de configuración para conocer cuántos parámetros se deben analizar del trazo y que clases contienen la implementación del analizador.

Las clases que queramos definir como analizadores deben implementar la interfaz *iKeyParameter*. Si cumple con esta interfaz, el sistema podrá a través de introspección invocar al método *quatize()* que será el encargado de fijar el valor para ese parámetro.

En primer lugar se definirán las variables que contendrán valores máximos, mínimos y se iniciará el cronómetro para analizar el rendimiento.

```
public Key createKey(ArrayList<Trazo> trazos){
  logger.debug("Iniciando fabricación de Llave");
  long inicioTs = Calendar.getInstance().getTimeInMillis();
  long min_value = 0;
  long max_value = 0;
  long avg_value = 0;

  ...
  ...
```

a continuación pasaremos a crear el array de parámetros que conformará las características de la llave creada

```
  ...

  ArrayList<iKeyParameter> array_parametros = new ArrayList<iKeyParameter>();
  Key key = new Key();

  ...
  ...
```

Posteriormente pasaremos a recorrer el array de parámetros que habremos obtenido del fichero de configuración *keyfactoryconfig.xml* y usaremos la propiedad *classname* definida en el fichero que nos indica que clase implementa el parámetro en cuestión.

Si el parámetro está activado según el fichero de configuración, se procederá a invocar al método *quantize()* que debe tener ya que implementa la interfaz *iKeyParameter*. Una vez invocado guardaremos el valor que definirá ese parámetro analizado.

El sistema puede configurarse para requerir que el usuario realice el mismo trazo varias veces tanto en la validación como en el registro de un trazo. Con esta medida el sistema gana en fiabilidad, ya que se quedará con el valor medio de todos los trazos realizados, quedando así amortiguadas las ligeras variaciones que se mostraban en la *Ilustración 21*: Superposición Trazado de este documento.

```
...

// para cada parametro
for(KeyFactoryParameterConfigBean keyParamAux : key_params_configurations){
  //Reseteo contadores
  min_value = 0;
  max_value = 0;
  avg_value = 0;
  if(keyParamAux.isEnabled()){
    for(Trazo trazo : trazos){
      Object parametro = Class.forName(keyParamAux.getClassName()).newInstance();
      Method m = parametro.getClass().getDeclaredMethod("quantize", Trazo.class);
      Long value = (Long)m.invoke(parametro,trazo);
      ...
      ...
      avg_value += value.longValue();
    }
    avg_value = avg_value/trazos.size();

    ...
    ...
```

Una vez obtenido el valor medio de este parámetro en los trazos repetidos por el usuario, sólo quedará crear un objeto que cumpla la interfaz iKeyParameter y fijarle los valores calculados para guardarlo en el array que formará la llave

```
...

  Object parametro = Class.forName(keyParamAux.getClassName()).newInstance();
  Method m = parametro.getClass().getDeclaredMethod("setParamName", String.class);
  m.invoke(parametro,keyParamAux.getName());
  m = parametro.getClass().getDeclaredMethod("setValue", Long.class);
  m.invoke(parametro,avg_value);
  array_parametros.add((iKeyParameter)parametro);
  ...
  ...
key.setParametros(array_parametros);
long finTs = Calendar.getInstance().getTimeInMillis();
logger.debug("Fin fabricación Llave ha tardado: "+(finTs-inicioTs)+" ms");
return key;
```

x. Paquete hda.auth.locks

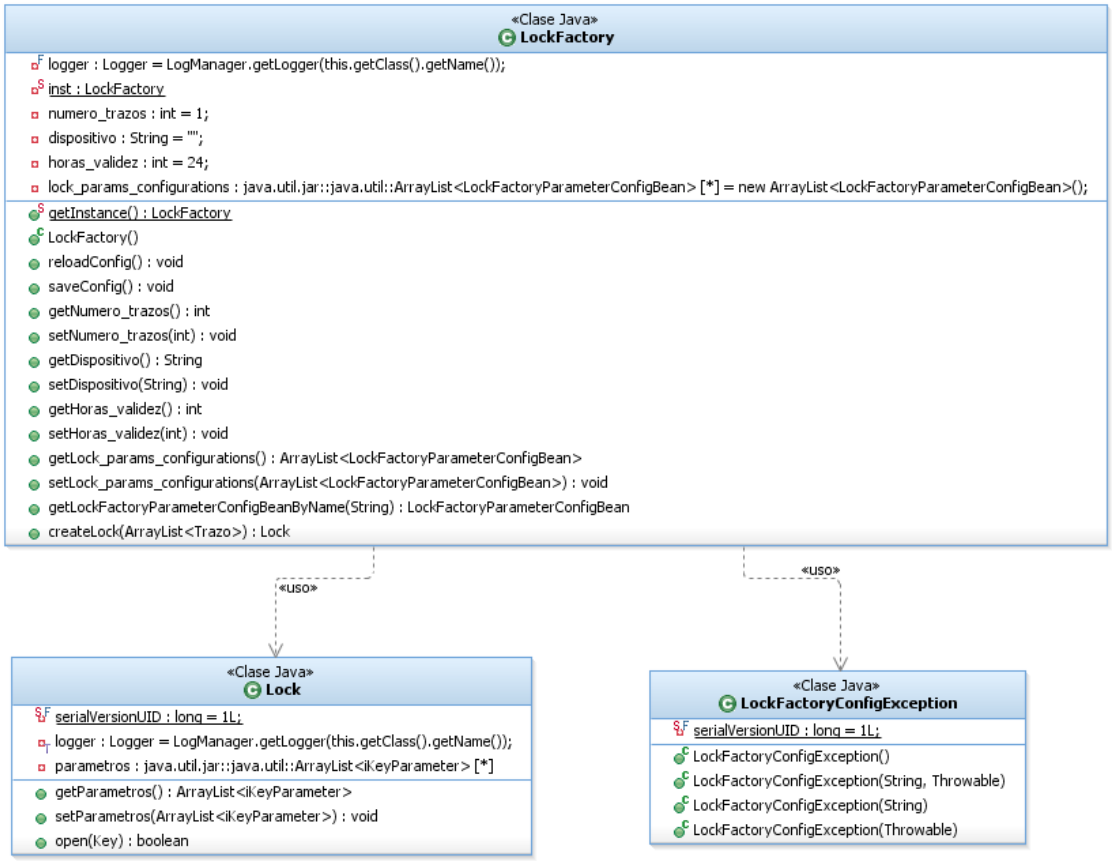


Ilustración 45: Paquete hda.auth.locks

Este paquete contiene las clases que intervienen en la fabricación de la Cerradura. Al igual que en el caso anterior, *LockFactory* está implementada como un singleton que lee su configuración de *lockfactoryconfig.xml* donde tiene las instrucciones de cómo debe proceder para crear un nuevo *Lock*.

La diferencia más reseñable respecto al caso anterior, es la aparición de un nuevo concepto del que ya se ha hablado en este documento y que hemos denominado Tolerancia.

En el método *createLock(ArrayList<Trazo>)* la mayor diferencia se encuentra en la obtención de los valores máximos y mínimos de un determinado parámetro entre los trazos que se analizan y en el cálculo de la tolerancia.

```
for(LockFactoryParameterConfigBean lockParamAux : lock_params_configurations){
  //Reseteo contadores
  min_value = 0;
  max_value = 0;
  avg_value = 0;
    if(lockParamAux.isEnabled()){
      for(Trazo trazo : trazos){
        Object parametro = Class.forName(lockParamAux.getClassName()).newInstance();
        Method m = parametro.getClass().getDeclaredMethod("quantize", Trazo.class);
        Long value = (Long)m.invoke(parametro,trazo);

        if( (min_value == 0) || (value.longValue()<min_value)){
          min_value = value.longValue();
        }
        if( (max_value == 0) || (value.longValue()>min_value)){
          max_value = value.longValue();
        }
        avg_value += value.longValue();

        ...
        ...

      avg_value = avg_value/trazos.size();

      long diferencia_max_avg=max_value - avg_value;
      long diferencia_avg_min=avg_value - min_value;
      if(diferencia_max_avg>diferencia_avg_min){
        tolerance = new Float(1)-( (new Float(avg_value)/new Float(max_value)));
      }else{
        tolerance = new Float(1)-( (new Float(min_value)/new Float(avg_value)));
      }
      int toleranceInt = new Float(tolerance*100).intValue();

      ...
      ...
```

La Tolerancia que vemos en estas líneas de código es la que hemos denominado Tolerancia Calculada, porque se basa en un porcentaje que expresa la máxima diferencia entre el valor extremo por arriba o por abajo del parámetro analizado, con relación a la media. Es una especie de valor máximo de la varianza que expresa las diferencias que el usuario de forma involuntaria realiza en ese parámetro al tratar de repetir el mismo trazo.

Al margen de esta Tolerancia Calculada, existe otra tolerancia denominada Tolerancia de Incertidumbre, que el administrador puede configurar a través del panel de control de la aplicación y que permite ajustar el funcionamiento de la aplicación a las necesidades particulares que cada usuario tenga. La tolerancia total será la suma de ambas tolerancias, la calculada y la de incertidumbre.

Como se ha dicho en varias ocasiones, la aplicación debe ser altamente parametrizable debido a que el usuario final cuenta con unas necesidades especiales que no se pueden prever de antemano y la herramienta debe ser configurada para cubrir todas las situaciones que pudieran surgir.

xi. Paquete hda.auth.parameters

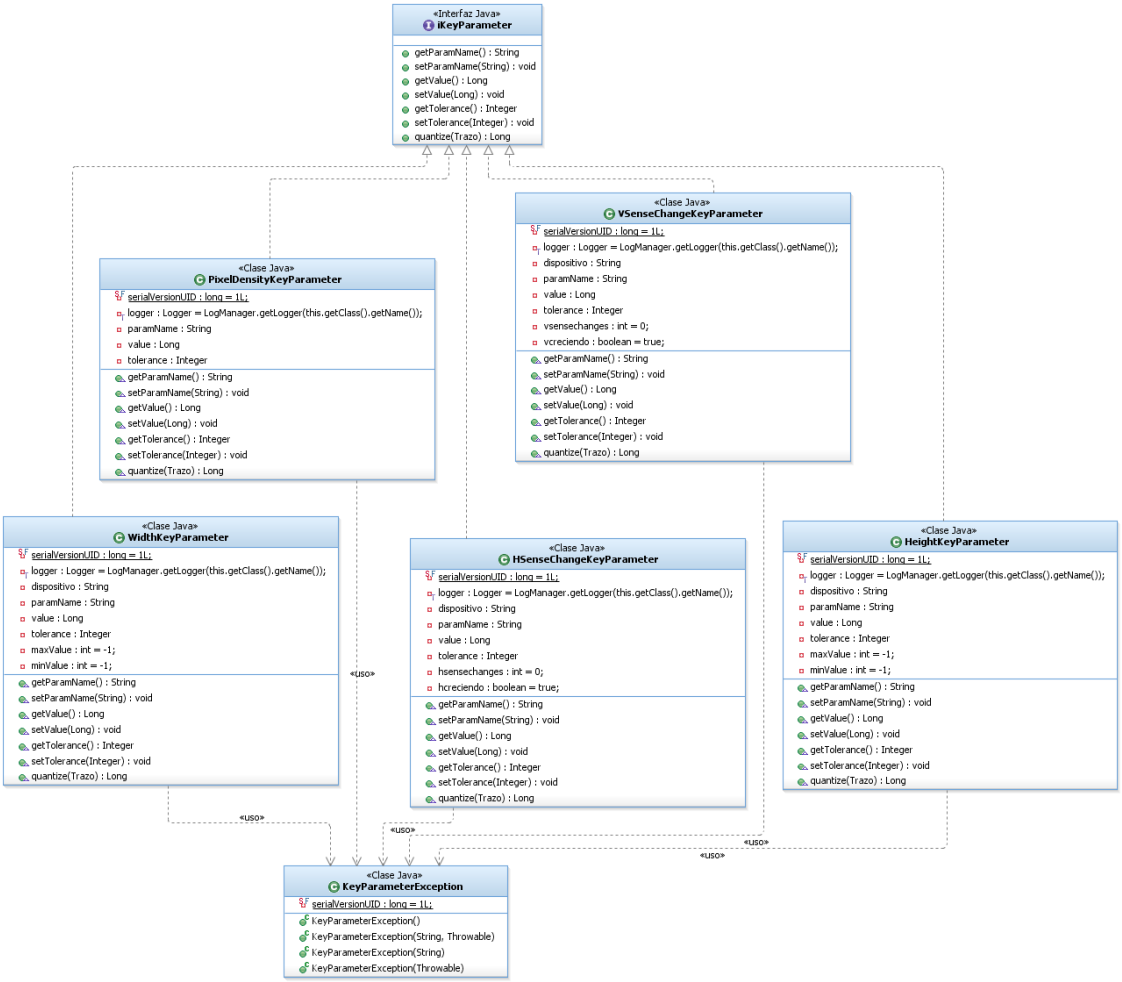


Ilustración 46: Paquete hda.auth.parameters

En este paquete se pueden encontrar los parámetros que se han implementado para la aplicación, si bien es cierto que es posible extender las bibliotecas con implementaciones propias.

Todas ellas implementan la interfaz *iKeyParameter* como puede apreciarse y hacen uso de una excepción propia para poder capturar posibles problemas a la hora de realizar los cálculos.

3.4. Elementos Configurables

La herramienta desarrollada, cuenta con multitud de configuraciones posibles. Se ha decidido crear un sistema altamente parametrizable debido a que debe ser capaz de adaptarse a las distintas necesidades de los usuarios.

Dado que se trata de un sistema de autenticación destinado al HDA, se debe prestar especial atención a la adaptabilidad que el sistema pueda ofrecer a las distintas situaciones que se puedan presentar.

3.4.1. Propiedades

Dado el elevado número de propiedades que componen los ficheros de configuración, se ha decidido crear unas estructuras XSD que limiten y a la vez ayuden a determinar las propiedades del sistema.

Cada módulo del sistema cuenta con un fichero de configuración principal que está escrito en XML y asociado a un determinado Schema XSD. El sistema en el arranque, comprobará que los parámetros de configuración cumplen con los requisitos de su fichero XSD asociado. Si los ficheros de configuración no son validados con éxito en el inicio, el sistema lanzará una excepción informando del error de configuración. El proceso de validación se realizará a través de la clase *org.xml.sax.XMLReader*

A continuación se muestran los distintos ficheros de configuración presentes así como su estructura XSD.

a) MainConfigSchema.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- definition of simple elements -->
     <xs:simpleType name="nametype">
        <xs:restriction base="xs:string">
           <xs:minLength value="5"></xs:minLength>
           <xs:maxLength value="100"></xs:maxLength>
        </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="versiontype">
        <xs:restriction base="xs:string">
           <xs:minLength value="3"></xs:minLength>
           <xs:maxLength value="100"></xs:maxLength>
           <xs:pattern value="\d.\d"></xs:pattern>
        </xs:restriction>
     </xs:simpleType>


     <xs:simpleType name="statustype">
        <xs:restriction base="xs:string">
           <xs:enumeration value="enabled"/>
           <xs:enumeration value="disabled"/>
         </xs:restriction>
     </xs:simpleType>

     <!-- definition of complex elements -->
     <xs:complexType name="moduletype">
        <xs:sequence>
           <xs:element name="name" type="nametype"/>
           <xs:element name="version" type="versiontype"/>
           <xs:element name="configfile" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="configtype">
      <xs:sequence>
        <xs:element name="module" maxOccurs="unbounded" type="moduletype"/>
      </xs:sequence>
    </xs:complexType>

    <!-- elements -->
    <xs:element name="config" type="configtype"/>


</xs:schema>
```

b) MainConfig.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./config/MainConfigSchema.xsd" >
    <module>
        <name>LockFactory</name>
        <version>1.0</version>
        <configfile>./config/lockfactoryconfig.xml</configfile>
    </module>
    <module>
        <name>KeyFactory</name>
        <version>1.0</version>
        <configfile>./config/keyfactoryconfig.xml</configfile>
    </module>
    <module>
        <name>Logger</name>
        <version>1.0</version>
        <configfile>./config/log4j2.xml</configfile>
    </module>
    <module>
        <name>LockObject</name>
        <version>1.0</version>
        <configfile>./data/lock.dat</configfile>
    </module>
    <module>
        <name>LoginObject</name>
        <version>1.0</version>
        <configfile>./data/login.xml</configfile>
    </module>
    <module>
        <name>PrivateKey</name>
        <version>1.0</version>
        <configfile>./data/private.key</configfile>
    </module>
    <module>
        <name>AdminPasswordHASH</name>
        <version>1.0</version>
        <configfile>./data/adminPassword.dat</configfile>
    </module>
    <module>
        <name>DirSchemas</name>
        <version>1.0</version>
        <configfile>./config</configfile>
    </module>
</config>
```

c) *LockFactoryConfigSchema.xsd*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- definition of simple elements -->
    <xs:simpleType name="numerotrazostype">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="5"></xs:maxInclusive>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="devicetype">
        <xs:restriction base="xs:string">
            <xs:enumeration value="udraw"/>
            <xs:enumeration value="chronosTI"/>
         </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="horasvalideztype">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="48"></xs:maxInclusive>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="nametype">
        <xs:restriction base="xs:string">
            <xs:minLength value="5"></xs:minLength>
            <xs:maxLength value="100"></xs:maxLength>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="classtype">
        <xs:restriction base="xs:string">
            <xs:minLength value="5"></xs:minLength>
            <xs:maxLength value="100"></xs:maxLength>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tolerancetype">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"></xs:minInclusive>
            <xs:maxInclusive value="100"></xs:maxInclusive>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="statustype">
        <xs:restriction base="xs:string">
            <xs:enumeration value="enabled"/>
            <xs:enumeration value="disabled"/>
         </xs:restriction>
    </xs:simpleType>
    <!-- definition of complex elements -->
    <xs:complexType name="parametertype">
        <xs:sequence>
            <xs:element name="name" type="nametype"/>
            <xs:element name="class" type="classtype"/>
            <xs:element name="tolerance" type="tolerancetype"/>
            <xs:element name="status" type="statustype"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="configtype">
      <xs:sequence>
        <xs:element name="numerotrazos"  minOccurs="1" maxOccurs="1"
type="numerotrazostype" nillable="false"/>
        <xs:element name="dispositivo"  minOccurs="1" maxOccurs="1" type="devicetype"
nillable="false"/>
        <xs:element name="horasvalidez"  minOccurs="1" maxOccurs="1"
type="horasvalideztype" nillable="false"/>
         <xs:element name="parameter" maxOccurs="unbounded" type="parametertype"/>
      </xs:sequence>
    </xs:complexType>
    <!-- elements -->
    <xs:element name="config" type="configtype"/>
</xs:schema>
```

d) LockFactoryConfig.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./config/LockFactoryConfigSchema.xsd">
    <numerotrazos>3</numerotrazos>
    <dispositivo>udraw</dispositivo>
    <horasvalidez>24</horasvalidez>
    <parameter>
        <name>PixelDensity</name>
        <class>hda.auth.parameters.PixelDensityKeyParameter</class>
        <tolerance>10</tolerance>
        <status>enabled</status>
    </parameter>
    <parameter>
        <name>HSenseChange</name>
        <class>hda.auth.parameters.HSenseChangeKeyParameter</class>
        <tolerance>10</tolerance>
        <status>enabled</status>
    </parameter>
    <parameter>
        <name>VSenseChange</name>
        <class>hda.auth.parameters.VSenseChangeKeyParameter</class>
        <tolerance>10</tolerance>
        <status>enabled</status>
    </parameter>
    <parameter>
        <name>Height</name>
        <class>hda.auth.parameters.HeightKeyParameter</class>
        <tolerance>10</tolerance>
        <status>enabled</status>
    </parameter><parameter>
        <name>Width</name>
        <class>hda.auth.parameters.WidthKeyParameter</class>
        <tolerance>10</tolerance>
        <status>enabled</status>
    </parameter>
</config>
```

e) KeyFactoryConfigSchema.xsd

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- definition of simple elements -->
    <xs:simpleType name="numeroTrazostype">
       <xs:restriction base="xs:decimal">
          <xs:minInclusive value="1"></xs:minInclusive>
          <xs:maxExclusive value="5"></xs:maxExclusive>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="nametype">
       <xs:restriction base="xs:string">
          <xs:minLength value="5"></xs:minLength>
          <xs:maxLength value="100"></xs:maxLength>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="classtype">
       <xs:restriction base="xs:string">
          <xs:minLength value="5"></xs:minLength>
          <xs:maxLength value="100"></xs:maxLength>
       </xs:restriction>
     </xs:simpleType>


     <xs:simpleType name="statustype">
       <xs:restriction base="xs:string">
          <xs:enumeration value="enabled"/>
          <xs:enumeration value="disabled"/>
        </xs:restriction>
     </xs:simpleType>

     <!-- definition of complex elements -->
     <xs:complexType name="parametertype">
       <xs:sequence>
          <xs:element name="name" type="nametype"/>
          <xs:element name="class" type="classtype"/>
          <xs:element name="status" type="statustype"/>
       </xs:sequence>
    </xs:complexType>

    <xs:complexType name="configtype">
      <xs:sequence>
       <xs:element name="numerotrazos" minOccurs="1" maxOccurs="1"
type="numeroTrazostype" nillable="false"/>
        <xs:element name="parameter" maxOccurs="unbounded"
type="parametertype"/>
      </xs:sequence>
    </xs:complexType>

    <!-- elements -->
    <xs:element name="config" type="configtype"/>


</xs:schema>
```

f) KeyFactoryConfig.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./config/KeyFactoryConfigSchema.xsd">
  <numerotrazos>1</numerotrazos>
  <parameter>
    <name>PixelDensity</name>
    <class>hda.auth.parameters.PixelDensityKeyParameter</class>
    <status>enabled</status>
  </parameter>
  <parameter>
    <name>HSenseChange</name>
    <class>hda.auth.parameters.HSenseChangeKeyParameter</class>
    <status>enabled</status>
  </parameter>
  <parameter>
    <name>VSenseChange</name>
    <class>hda.auth.parameters.VSenseChangeKeyParameter</class>
    <status>enabled</status>
  </parameter>
  <parameter>
    <name>Height</name>
    <class>hda.auth.parameters.HeightKeyParameter</class>
    <status>enabled</status>
  </parameter>
  <parameter>
    <name>Width</name>
    <class>hda.auth.parameters.WidthKeyParameter</class>
    <status>enabled</status>
  </parameter>
</config>
```

g) log4j.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="error" monitorInterval="30">
    <Appenders>
        <File name="log" append="true" fileName="./logs/hdaAuth.log" >
            <PatternLayout pattern="%d [%t] %-5p %c{5} - %m%n" />
        </File>
        <File name="access" append="true" fileName="./logs/access_log.xml" >
            <PatternLayout pattern="%m%n" />
        </File>
        <Console name="STDOUT" target="SYSTEM_OUT" >
            <PatternLayout pattern="%d %-5p [%t] %C{6} (%F:%L) - %m%n" />
        </Console>
    </Appenders>
    <Loggers>
        <Logger name="hda.auth" level="fatal" >
            <AppenderRef ref="STDOUT" />
        </Logger>
        <Logger name="hda.auth.locks.Lock" level="info" >
            <AppenderRef ref="access" />
        </Logger>
        <Root level="debug" >
            <AppenderRef ref="log" />
        </Root>
    </Loggers>
</Configuration>
```

h) LoginSchema.xsd

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- definition of simple elements -->
    <xs:simpleType name="usertype">
       <xs:restriction base="xs:string">
          <xs:minLength value="5"></xs:minLength>
          <xs:maxLength value="100"></xs:maxLength>
       </xs:restriction>
    </xs:simpleType>

    <!-- definition of complex elements -->
     <xs:complexType name="valideztype">
       <xs:sequence>
          <xs:element name="noAntes" type="xs:long"/>
          <xs:element name="noDespues" type="xs:long"/>
       </xs:sequence>
    </xs:complexType>

    <xs:complexType name="accesotype">
      <xs:sequence>
       <xs:element name="user" minOccurs="1" maxOccurs="1" type="usertype"
nillable="false"/>
        <xs:element name="validez" minOccurs="1" maxOccurs="1"
type="valideztype" nillable="false"/>
        <xs:element name="firma" minOccurs="1" maxOccurs="1" type="xs:string"
nillable="false"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="logintype">
      <xs:sequence>
        <xs:element name="acceso" maxOccurs="unbounded" type="accesotype"/>
      </xs:sequence>
    </xs:complexType>

    <!-- elements -->
    <xs:element name="login" type="logintype" />

</xs:schema>
```

3.4.2. Bibliotecas

Las bibliotecas que se han empleado en la aplicación, se encuentran bajo el directorio PFG/lib como puede verse en la siguiente ilustración:

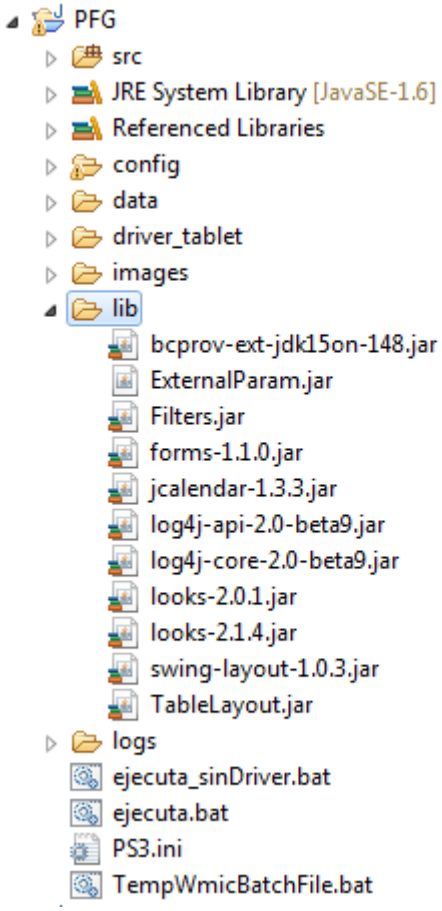


Ilustración 47: Bibliotecas empleadas

Existe la posibilidad de definir nuevas bibliotecas de análisis de parámetros. Para ellos será necesario incluir el nuevo parámetro en los ficheros keyfactoryconfig.xml y lockfactoryconfig.xml y a su vez incluir el código de la clase en la ruta del classpath. Por ejemplo si agrupásemos unas nuevas clases de análisis en el jar *ExternalParam.jar*, sería necesario incluir "*java -cp lib/ExternalParam.jar"* para que el entorno fuese capaz de encontrar la clase y poderla emplear.

3.4.3. Equipos

Los equipos deben configurarse con versiones de java 1.6 o superior. El equipo con la versión más baja sobre la que se ha probado con resultado satisfactorio era:

```
java version "1.6.0_26"
Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
Java HotSpot(TM) 64-Bit Server VM (build 20.1-b02, mixed mode)
```

No es necesario ningún tipo de configuración adicional para que la aplicación funcione de forma correcta. No obstante sería recomendable realizar cierta configuración adicional que permitiese aprovechar las ventajas de la seguridad ofrecida por el S.O que limitase el acceso a los ficheros de configuración otorgando sólo permisos de lectura y escritura al propietario chmod 700 y crear un usuario específico para la aplicación de autenticación que fuese el owner de dichos ficheros.

3.4.4. Otros Elementos

Para la comunicación con el uDraw de PS3, es necesario emplear un driver desarrollado por Brandon Wilson cuya configuración se puede encontrar en (14). Se ha percibido que a pesar de que la sensibilidad del lápiz es un parámetro modificable en el driver del desarrollador, no se ha conseguido que la sensación de escritura fuese fluida y natural, cosa que si se ha experimentado usando otro tipo de tabletas digitalizadoras como es la Bamboo de Wacom que también permite conectarla un adaptador que la dote de capacidades wireless.

Capítulo 4: Resultados de Verificación

4.1. Objetivos de calidad

Para garantizar el poder alcanzar los objetivos de calidad del sistema, se han empleado metodologías recogidas en CMMI. En concreto han sido aplicadas, en la medida de lo posible, algunas de las actividades recogidas en *Process and Product Qualty Assurance* (PPQA), un área de proceso de soporte en nivel de madurez 2 (17).

PPQA involucra aspectos como la evaluación objetiva de los procesos, productos y servicios en referencia a descripciones de procesos, estándares y procedimientos.

En esta ocasión nos centraremos únicamente en las labores de Validación de Requerimientos, ya que establecer planes de mejora y controles del cambio y otro tipo de procesos del ciclo de vida del producto, quedan fuera del alcance debido a que en principio, el ciclo de vida de este producto se parará después del prototipado y demostración.

4.1.1. Casos de Prueba

Para la validación de los requerimientos planteados, se han definido una serie de casos de prueba que deben ser superados para conseguir la aceptación del producto.

1) **Registrar Trazo de forma correcta**

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder al registro de un trazo en el sistema	Rol: Usuario	El sistema deberá guardar el trazo realizado y mostrar la opción de validación	✓ El sistema guarda el trazo realizado y pasa el botón de validar de inactivo a activo

2) **Registrar Trazo de forma incorrecta**

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder al registro de un trazo en el sistema y en mitad de la operación cerrar la ventana	Rol: Usuario	El sistema deberá continuar la ejecución normal sin guardar el trazo.	✓ El sistema al cerrar la ventana continua con la ejecución normal mostrando el botón validar inactivo.

3) Validar Trazo de forma correcta con resultado OK

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder a validar un trazo de forma correcta	Rol: Usuario	El sistema deberá indicar que el proceso ha sido correcto y deberá generar el token firmado	✓ El sistema muestra un icono de OK junto al botón validar y se genera un token

4) Validar Trazo de forma correcta y resultado NOK

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder a validar un trazo de forma correcta pero que no supere la validación	Rol: Usuario	El sistema deberá indicar que el proceso no ha superado la prueba y no se generará Token	✓ El sistema muestra un icono de NOK junto al botón validar

5) Validar Trazo de forma incorrecta

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder a validar un trazo y en mitad de la operación cerrar la ventana	Rol: Usuario	El sistema deberá continuar con la ejecución normal	✓ El sistema al cerrar la ventana continua operando con normalidad

6) Realizar Logout

Acción	Probador	Resultado esperado	Resultado obtenido
El usuario deberá proceder a realizar logout del sistema	Rol: Usuario	El sistema deberá eliminar el token e informar de ello al usuario	✓ El sistema indica a través de diálogos que el token se ha eliminado y efectivamente se ha borrado del sistema de ficheros

7) Generar Certificados de forma correcta

Acción	Probador	Resultado esperado	Resultado obtenido
El administrador deberá generar una pareja de claves mediante RSA	Rol: Administrador	El sistema deberá guardar en el sistema de ficheros la pareja de claves RSA generada	✓ El sistema pide una ubicación donde guardar la clave pública y la genera de forma correcta

8) Generar Certificados de forma incorrecta

Acción	Probador	Resultado esperado	Resultado obtenido
El administrador generará una pareja de claves RSA con un nombre de fichero inválido	Rol: Administrador	El sistema deberá informar que el proceso de generación ha fallado	✓ Al escoger un nombre no válido, el sistema indica que el proceso de generación falló

9) Cambiar Contraseña Administrador de forma correcta

Acción	Probador	Resultado esperado	Resultado obtenido
El administrador tras identificarse procederá a cambiar la contraseña de administrador	Rol: Administrador	El sistema indicará que la clave ha sido cambiada y a partir de este momento será válida para identificarse como administrador	✓ El sistema cambia de forma correcta la contraseña de administrador e informa por pantalla de ello

10) Cambiar Contraseña Administrador de forma incorrecta

Acción	Probador	Resultado esperado	Resultado obtenido
Al cambiar la contraseña de administrador intentar poner una clave distinta donde dice repetir clave.	Rol: Administrador	El sistema indicará que la clave debe repetirse para llevar a cabo el cambio	✓ El sistema no permite el cambio, informando de ello al administrador

11) Configurar parámetros del sistema

Acción	Probador	Resultado esperado	Resultado obtenido
El administrador tras identificarse intentará cambiar una serie de parámetros	Rol: Administrador	El sistema deberá reflejar los cambios de configuración realizados por el administrador	✓ El sistema responde correctamente ante los cambios de configuración

12) Comprobar persistencia de datos

Acción	Probador	Resultado esperado	Resultado obtenido
Cerrar la aplicación y volverla abrir para comprobar persistencia de datos	Rol: Administrador	El sistema debe arrancar con las configuraciones y datos de la última vez que se cerró	✓ El sistema conserva los cambios realizados después del reinicio de la aplicación

13) Comprobar firma token

Acción	Probador	Resultado esperado	Resultado obtenido
Modificar algún valor del token y comprobar la firma con la herramienta de PFG_EXTRAS	Rol: Administrador	La herramienta para comprobar la firma debe detectar que el token ha sido modificado	✓ La herramienta muestra los campos del token e indica de forma clara que la firma no es válida

14) Comprobar cifrado claves

Acción	Probador	Resultado esperado	Resultado obtenido
Comprobar que la clave privada está cifrada con una clave secreta al guardarla en el sistema de ficheros	Rol: Administrador	La clave privada no podrá usarse junto con la clave pública sin antes descifrarla mediante AES y la clave secreta	✓ Para poder emplear la clave privada es necesario desencriptarla con la clave secreta del sistema y el algoritmo AES

4.1.2. Pruebas unitarias

Las pruebas unitarias son pruebas de testing que se realizan sobre elementos o módulos individuales o unidades relacionadas.

Las pruebas unitarias que se han realizado en el desarrollo de este proyecto inicialmente fueron hechas mediante JUnit, una herramienta proporcionada por java para la realización de pruebas intensivas y análisis de resultados.

No obstante, como era requerido en casi todas las pruebas la intervención del componente humano debido a que era necesario estar continuamente realizando trazos sobre la tableta, se consideró que el valor añadido de JUnit no era suficiente para justificar el trabajo extra que suponía diseñar las pruebas. Por ello finalmente las pruebas unitarias se han ido realizando de forma manual durante el desarrollo.

4.1.3. Pruebas Concepto

Las pruebas de concepto son una serie de pruebas que se han realizado para comprobar que la aplicación se comportaba de forma correcta.

A continuación se ofrecen una batería de pruebas y los valores obtenidos para ellas que serán analizados posteriormente.

En el primer ejemplo que se muestra se puede ver el registro de un trazo se corresponde con la letra Z realizado 3 veces. Como se puede apreciar, aunque nuestra intención haya sido realizar el mismo trazo, siempre existen ligeras variaciones, es aquí donde el sistema entra a calcular los valores medios de los parámetros y las tolerancias calculadas re representan una especie de varianza máxima que será muy importante de cara a optimizar el funcionamiento de la herramienta.

Existe otro factor del cual ya se ha hablado en este documento llamado tolerancia de incertidumbre, que es un valor configurable a través de la aplicación y que nos permite establecer una tolerancia añadida a la calculada independiente para cada parámetro. Así podremos hacer que nuestro sistema sea muy estricto con la validación o un poco más flexible si las necesidades del usuario requieren que así sea.

A continuación se muestra una captura con el caso de prueba del cual estamos hablando, el cual finalmente se han superpuesto los 3 trazados para ver claramente a qué nos estamos refiriendo cuando hablamos de ligeras variaciones.

Las pruebas que se han realizado han sido todas ellas hechas con la interfaz en forma de tableta digitalizadora, no pudiéndose comprobar el funcionamiento del sistema con el periférico en forma de reloj de Texas Instruments, debido a que a pesar de estar la herramienta preparada, no se pudo contar finalmente con él.

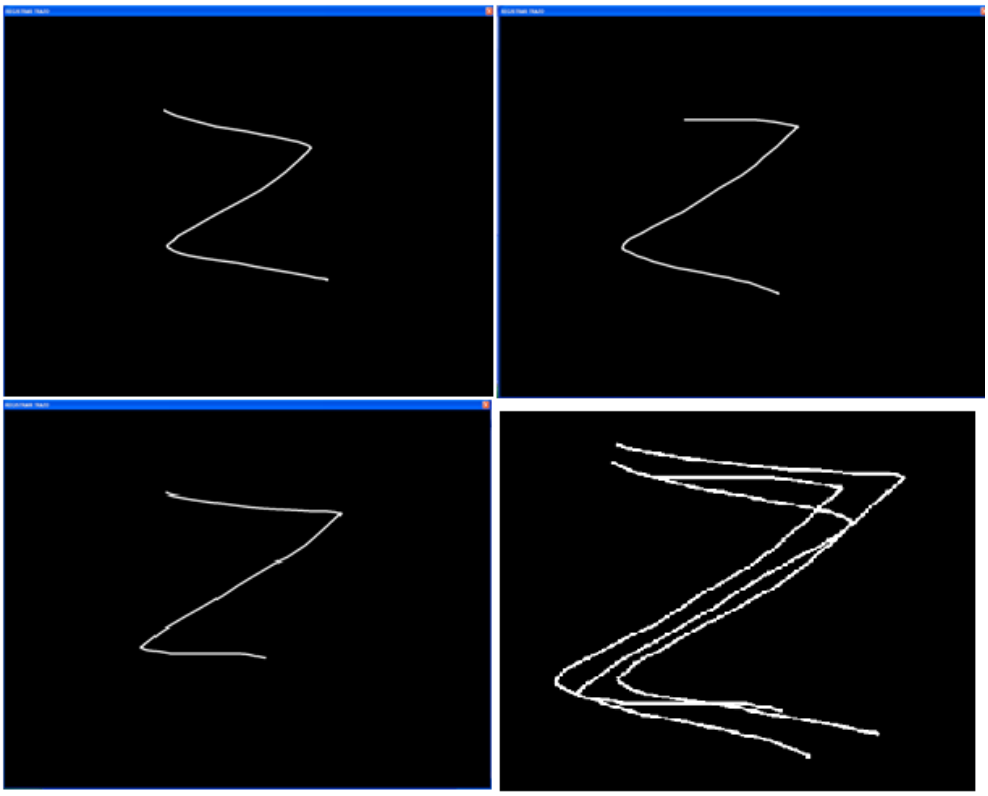


Ilustración 48: Caso de prueba 1

Parametro<PixelDensity>
Valor medio: 20208
Tolerancia calculada: 6 %
Tolerancia incertidumbre:10 %
Tolerancia Total:16 %
Parametro<HSenseChange>
Valor medio: 3
Tolerancia calculada: 25 %
Tolerancia incertidumbre:10 %
Tolerancia Total:35 %
Parametro<VSenseChange>
Valor medio: 0
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 442
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %
Parametro<Width>
Valor medio: 467
Tolerancia calculada: 10 %
Tolerancia incertidumbre:10 %
Tolerancia Total:20 %

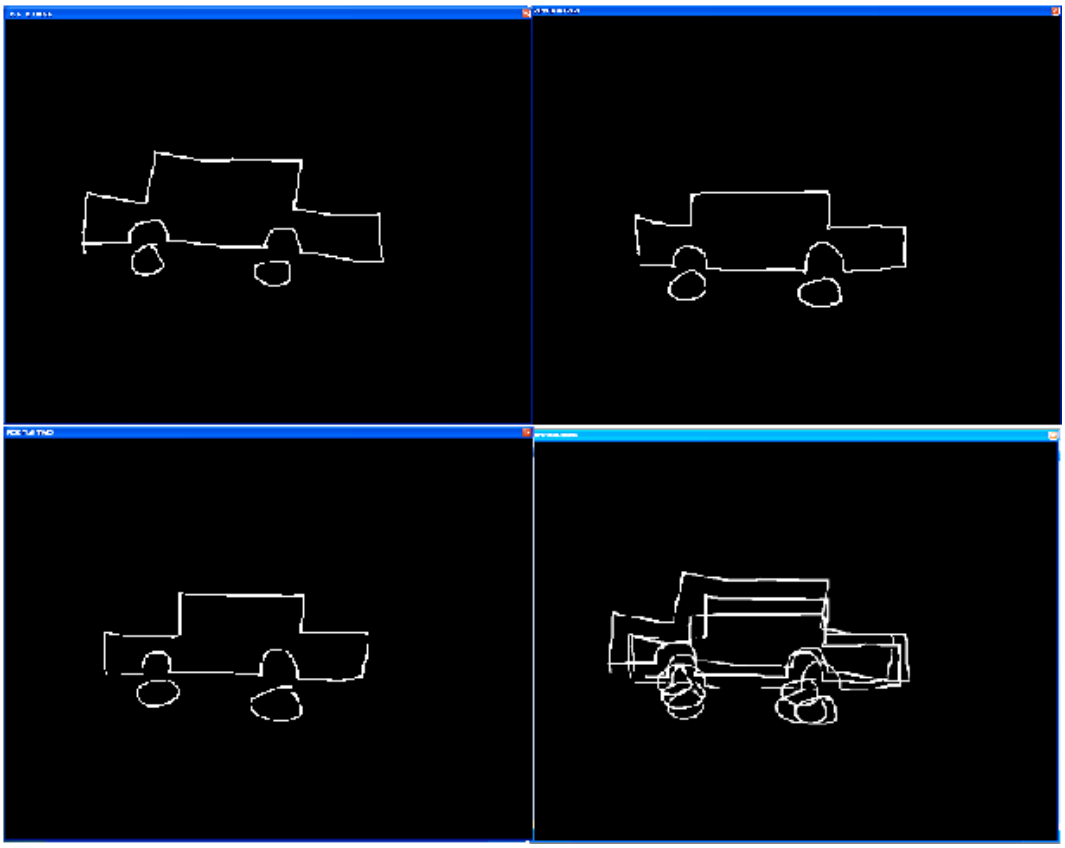


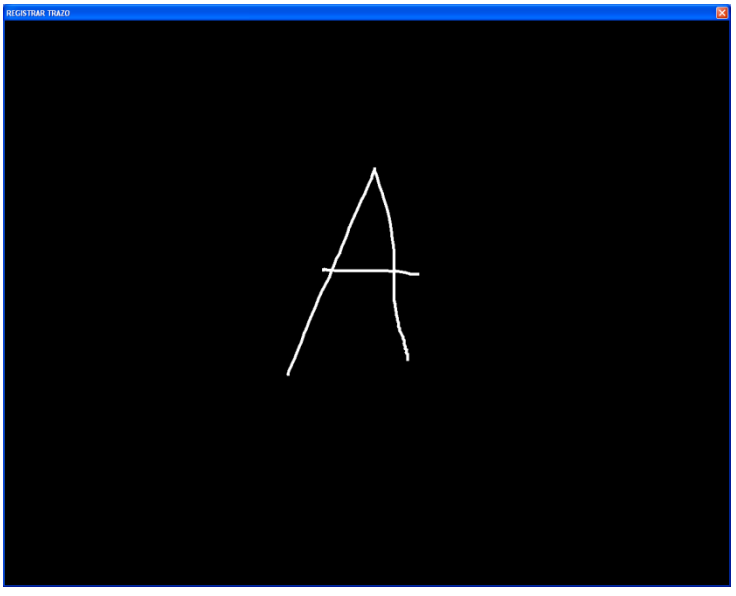
Ilustración 49: Caso de prueba 2

Parametro<PixelDensity>
Valor medio: 37649
Tolerancia calculada: 4 %
Tolerancia incertidumbre:10 %
Tolerancia Total:14 %
Parametro<HSenseChange>
Valor medio: 20
Tolerancia calculada: 10 %
Tolerancia incertidumbre:10 %
Tolerancia Total:20 %
Parametro<VSenseChange>
Valor medio: 17
Tolerancia calculada: 19 %
Tolerancia incertidumbre:10 %
Tolerancia Total:29 %
Parametro<Height>
Valor medio: 306
Tolerancia calculada: 10 %
Tolerancia incertidumbre:10 %
Tolerancia Total:20 %
Parametro<Width>
Valor medio: 670
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %

Después de poner como ejemplo dos casos de prueba uno más simple y otro ligeramente más complejo superpuestos, un primer dato que nos llama la atención es la tolerancia calculada en el caso de prueba 1 para el parámetro HSenseChange que es del 25%, es un valor bastante alto para lo simple que es el trazo. Esto es debido a que el valor de la media es muy bajo, en concreto 3 y una simple variación en alguno de los 3 trazos, es decir que en alguno de ello el valor de ese parámetro sea 4, hace que rápidamente la varianza crezca al 25%.

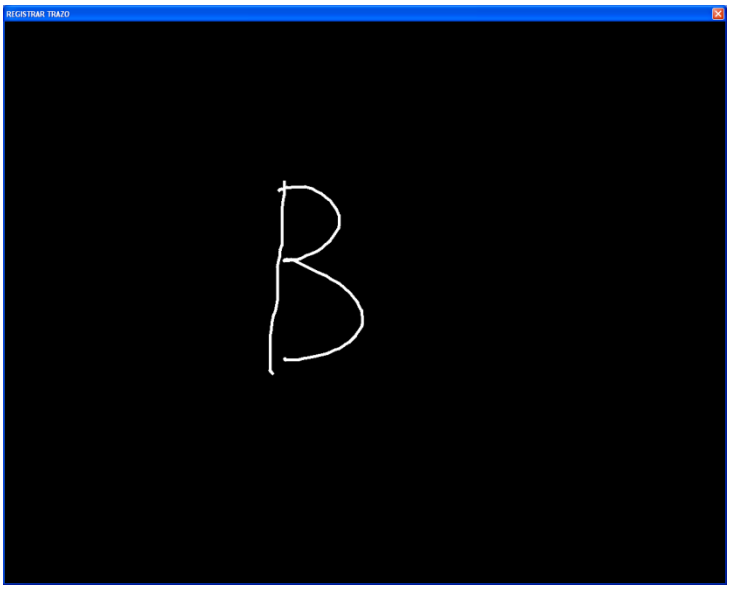
Con este análisis una primera premisa nos indica que los trazos demasiado simples son inestables y hacen que en algunos casos la tolerancia se dispare, haciendo el sistema más vulnerable. A este fenómeno de valores de media bajos con tolerancias altas le hemos denominado holgura, ya que hacen el sistema más permisivo. Una situación ideal mantendrá los valores de tolerancia estables en torno al 10-20%

A continuación se mostrarán distintos casos de prueba con sólo una de las capturas aunque en todos los casos el trazo fue realizado 3 veces. Se han realizado pruebas con letras, dibujos simples, formas geométricas y grupos de formas geométricas.



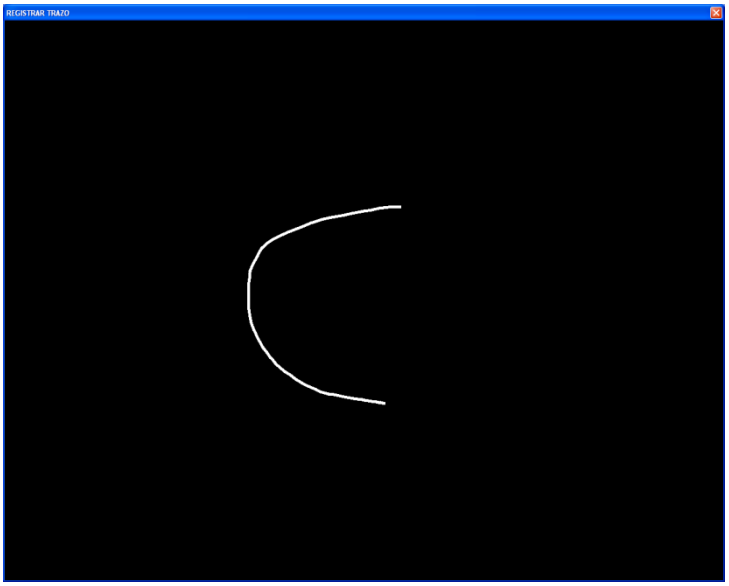
Parametro<PixelDensity>
Valor medio: 12155
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %
Parametro<HSenseChange>
Valor medio: 3
Tolerancia calculada: 39 %
Tolerancia incertidumbre:10 %
Tolerancia Total:49 %
Parametro<VSenseChange>
Valor medio: 4
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 308
Tolerancia calculada: 14 %
Tolerancia incertidumbre:10 %
Tolerancia Total:24 %
Parametro<Width>
Valor medio: 198
Tolerancia calculada: 15 %
Tolerancia incertidumbre:10 %
Tolerancia Total:25 %

Ilustración 50: Caso de prueba 3



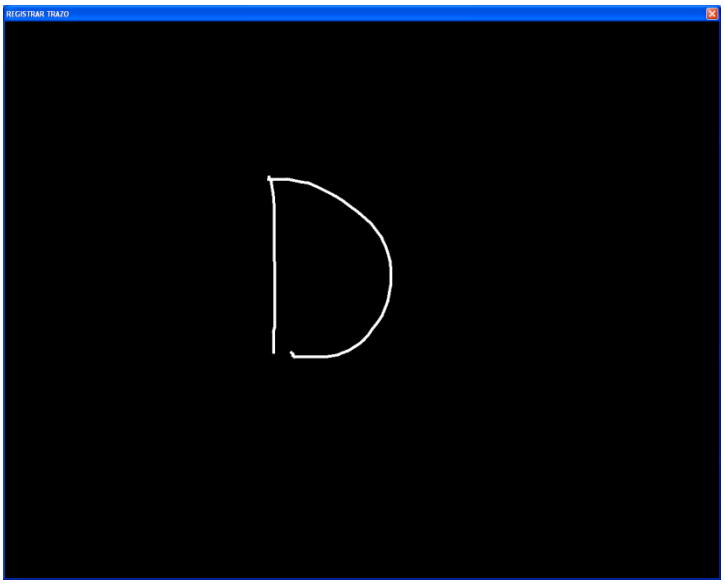
Parametro<PixelDensity>
Valor medio: 14712
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %
Parametro<HSenseChange>
Valor medio: 6
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<VSenseChange>
Valor medio: 3
Tolerancia calculada: 39 %
Tolerancia incertidumbre:10 %
Tolerancia Total:49 %
Parametro<Height>
Valor medio: 328
Tolerancia calculada: 6 %
Tolerancia incertidumbre:10 %
Tolerancia Total:16 %
Parametro<Width>
Valor medio: 162
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %

Ilustración 51: Caso de prueba 4



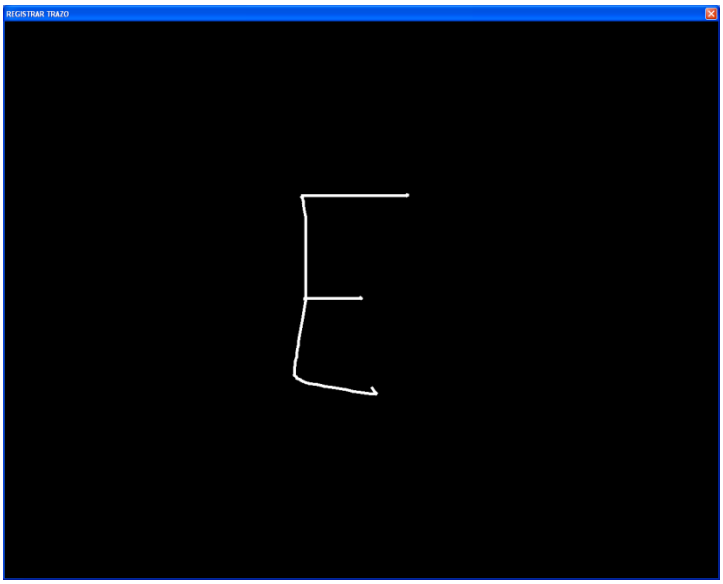
Parametro<PixelDensity>
Valor medio: 10395
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<HSenseChange>
Valor medio: 3
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<VSenseChange>
Valor medio: 0
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 345
Tolerancia calculada: 4 %
Tolerancia incertidumbre:10 %
Tolerancia Total:14 %
Parametro<Width>
Valor medio: 270
Tolerancia calculada: 10 %
Tolerancia incertidumbre:10 %
Tolerancia Total:20 %

Ilustración 52: Caso de prueba 5



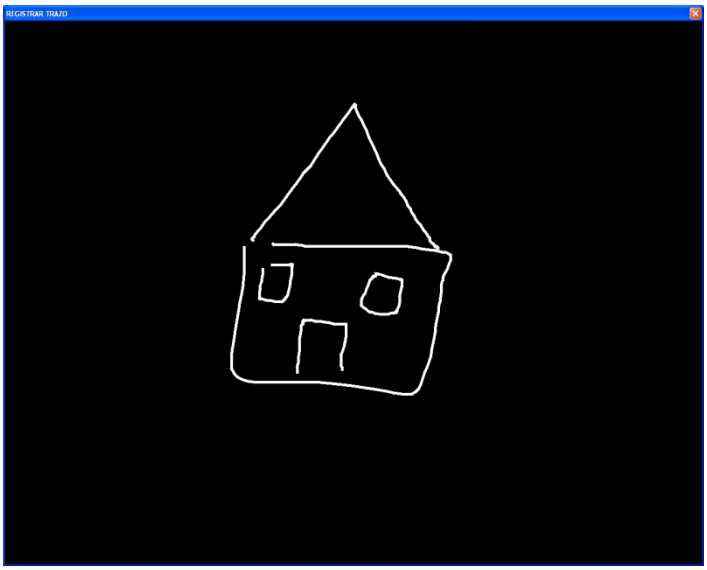
Parametro<PixelDensity>
Valor medio: 12528
Tolerancia calculada: 4 %
Tolerancia incertidumbre:10 %
Tolerancia Total:14 %
Parametro<HSenseChange>
Valor medio: 4
Tolerancia calculada: 19 %
Tolerancia incertidumbre:10 %
Tolerancia Total:29 %
Parametro<VSenseChange>
Valor medio: 3
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 302
Tolerancia calculada: 4 %
Tolerancia incertidumbre:10 %
Tolerancia Total:14 %
Parametro<Width>
Valor medio: 192
Tolerancia calculada: 10 %
Tolerancia incertidumbre:10 %
Tolerancia Total:20 %

Ilustración 53: Caso de prueba 6



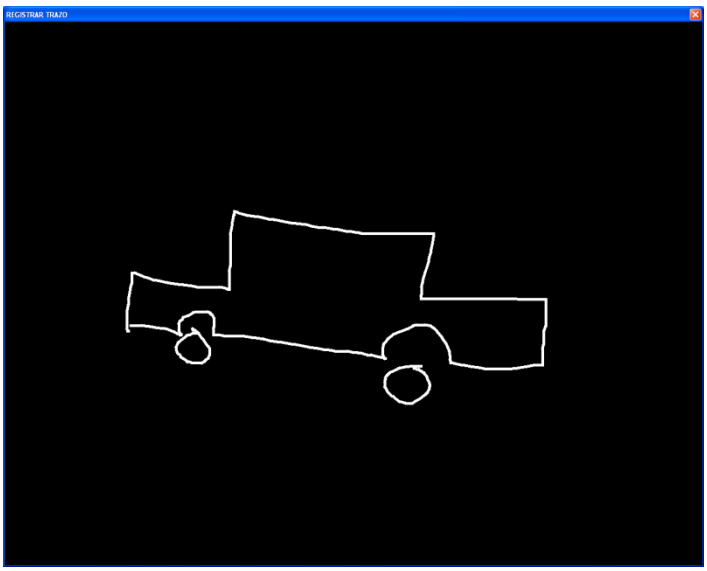
Parametro<PixelDensity>
Valor medio: 11591
Tolerancia calculada: 3 %
Tolerancia incertidumbre:10 %
Tolerancia Total:13 %
Parametro<HSenseChange>
Valor medio: 7
Tolerancia calculada: 12 %
Tolerancia incertidumbre:10 %
Tolerancia Total:22 %
Parametro<VSenseChange>
Valor medio: 2
Tolerancia calculada: 50 %
Tolerancia incertidumbre:10 %
Tolerancia Total:60 %
Parametro<Height>
Valor medio: 338
Tolerancia calculada: 4 %
Tolerancia incertidumbre:10 %
Tolerancia Total:14 %
Parametro<Width>
Valor medio: 202
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %

Ilustración 54: Caso de prueba 7



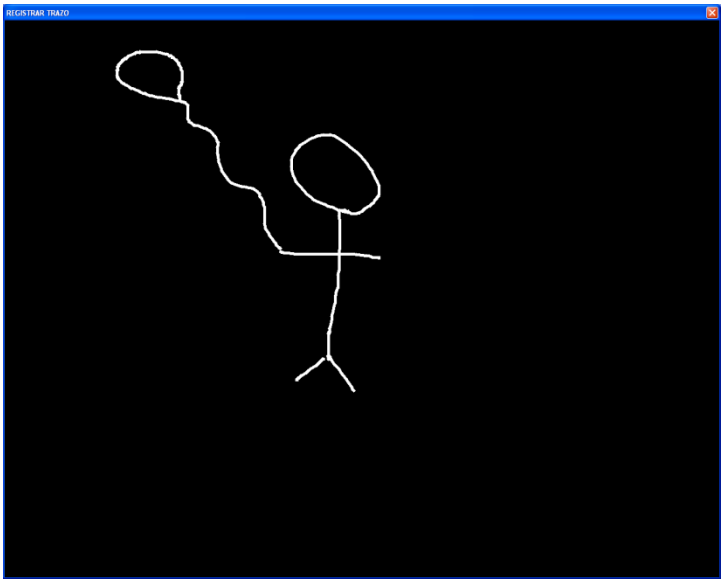
Parametro<PixelDensity>
Valor medio: 37310
Tolerancia calculada: 22 %
Tolerancia incertidumbre:10 %
Tolerancia Total:32 %
Parametro<HSenseChange>
Valor medio: 15
Tolerancia calculada: 16 %
Tolerancia incertidumbre:10 %
Tolerancia Total:26 %
Parametro<VSenseChange>
Valor medio: 11
Tolerancia calculada: 26 %
Tolerancia incertidumbre:10 %
Tolerancia Total:36 %
Parametro<Height>
Valor medio: 513
Tolerancia calculada: 17 %
Tolerancia incertidumbre:10 %
Tolerancia Total:27 %
Parametro<Width>
Valor medio: 374
Tolerancia calculada: 26 %
Tolerancia incertidumbre:10 %
Tolerancia Total:36 %

Ilustración 55: Caso de prueba 8



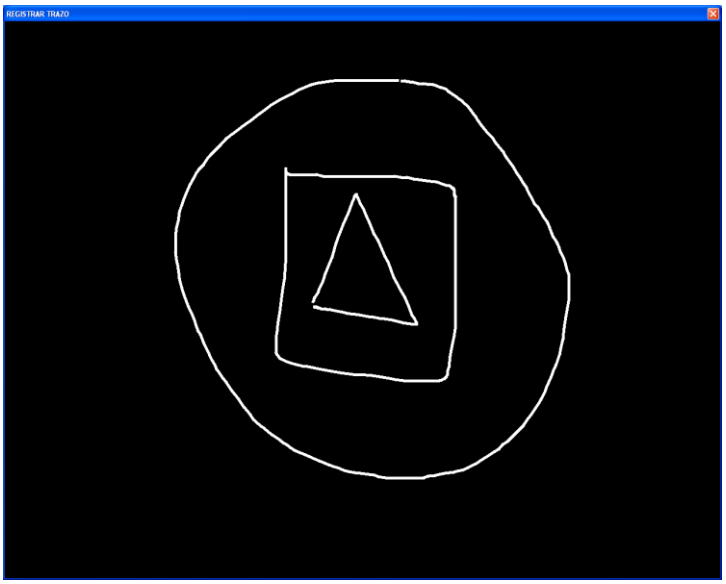
Parametro<PixelDensity>
Valor medio: 39054
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %
Parametro<HSenseChange>
Valor medio: 18
Tolerancia calculada: 22 %
Tolerancia incertidumbre:10 %
Tolerancia Total:32 %
Parametro<VSenseChange>
Valor medio: 16
Tolerancia calculada: 18 %
Tolerancia incertidumbre:10 %
Tolerancia Total:28 %
Parametro<Height>
Valor medio: 318
Tolerancia calculada: 6 %
Tolerancia incertidumbre:10 %
Tolerancia Total:16 %
Parametro<Width>
Valor medio: 719
Tolerancia calculada: 6 %
Tolerancia incertidumbre:10 %
Tolerancia Total:16 %

Ilustración 56: Caso de prueba 9



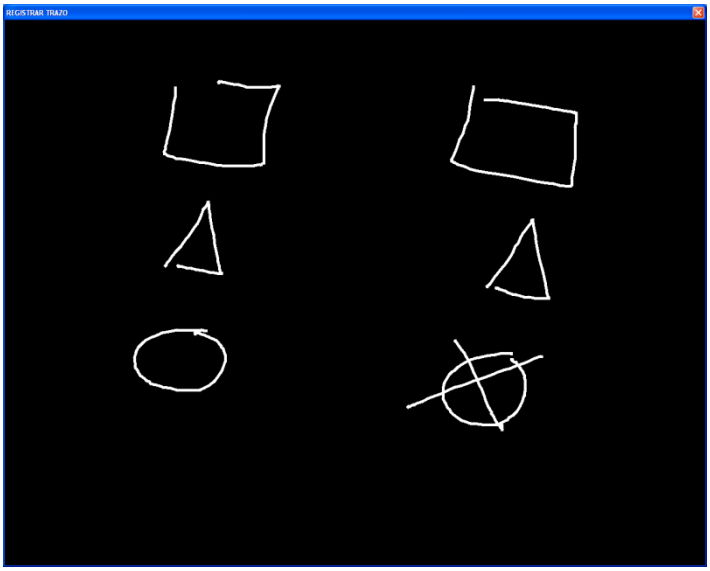
Parametro<PixelDensity>
Valor medio: 27516
Tolerancia calculada: 1 %
Tolerancia incertidumbre:10 %
Tolerancia Total:11 %
Parametro<HSenseChange>
Valor medio: 14
Tolerancia calculada: 17 %
Tolerancia incertidumbre:10 %
Tolerancia Total:27 %
Parametro<VSenseChange>
Valor medio: 9
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<Height>
Valor medio: 636
Tolerancia calculada: 5 %
Tolerancia incertidumbre:10 %
Tolerancia Total:15 %
Parametro<Width>
Valor medio: 479
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %

Ilustración 57: Caso de prueba 10



Parametro<PixelDensity>
Valor medio: 68558
Tolerancia calculada: 12 %
Tolerancia incertidumbre:10 %
Tolerancia Total:22 %
Parametro<HSenseChange>
Valor medio: 10
Tolerancia calculada: 28 %
Tolerancia incertidumbre:10 %
Tolerancia Total:38 %
Parametro<VSenseChange>
Valor medio: 7
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 765
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<Width>
Valor medio: 729
Tolerancia calculada: 9 %
Tolerancia incertidumbre:10 %
Tolerancia Total:19 %

Ilustración 58: Caso de prueba 11



Parametro<PixelDensity>
Valor medio: 52065
Tolerancia calculada: 1 %
Tolerancia incertidumbre:10 %
Tolerancia Total:11 %
Parametro<HSenseChange>
Valor medio: 22
Tolerancia calculada: 9 %
Tolerancia incertidumbre:10 %
Tolerancia Total:19 %
Parametro<VSenseChange>
Valor medio: 22
Tolerancia calculada: 9 %
Tolerancia incertidumbre:10 %
Tolerancia Total:19 %
Parametro<Height>
Valor medio: 697
Tolerancia calculada: 9 %
Tolerancia incertidumbre:10 %
Tolerancia Total:19 %
Parametro<Width>
Valor medio: 790
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %

Ilustración 59: Caso de prueba 12

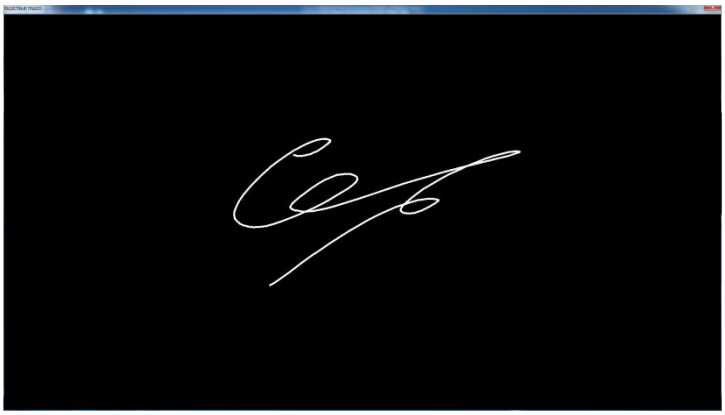
Las siguientes pruebas se han realizado en un equipo 16:9. Como puede apreciarse la superficie destinada a realizar el trazo se adapta al monitor con el que se esté trabajando.

Parametro<PixelDensity>
Valor medio: 56176
Tolerancia calculada: 13 %
Tolerancia incertidumbre:10 %
Tolerancia Total:23 %
Parametro<HSenseChange>
Valor medio: 27
Tolerancia calculada: 3 %
Tolerancia incertidumbre:10 %
Tolerancia Total:13 %
Parametro<VSenseChange>
Valor medio: 27
Tolerancia calculada: 3 %
Tolerancia incertidumbre:10 %
Tolerancia Total:13 %
Parametro<Height>
Valor medio: 484
Tolerancia calculada: 15 %
Tolerancia incertidumbre:10 %
Tolerancia Total:25 %
Parametro<Width>
Valor medio: 549
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %

Ilustración 60: Caso de prueba 13

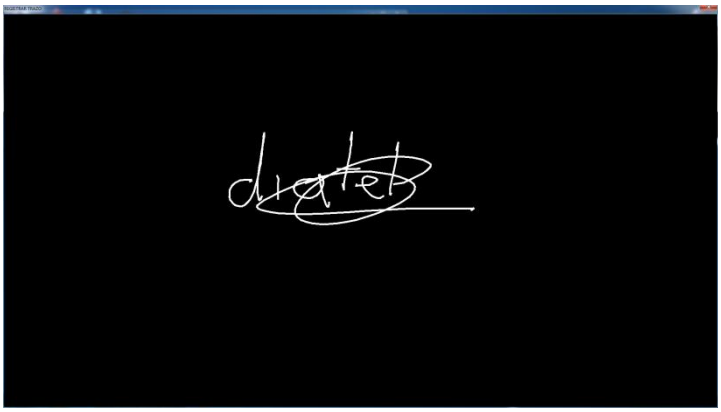
Parametro<PixelDensity>
Valor medio: 30674
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %
Parametro<HSenseChange>
Valor medio: 9
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<VSenseChange>
Valor medio: 9
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<Height>
Valor medio: 400
Tolerancia calculada: 2 %
Tolerancia incertidumbre:10 %
Tolerancia Total:12 %
Parametro<Width>
Valor medio: 694
Tolerancia calculada: 18 %
Tolerancia incertidumbre:10 %
Tolerancia Total:28 %

Ilustración 61: Caso de prueba 14



Parametro<PixelDensity>
Valor medio: 37166
Tolerancia calculada: 12 %
Tolerancia incertidumbre:10 %
Tolerancia Total:22 %
Parametro<HSenseChange>
Valor medio: 9
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<VSenseChange>
Valor medio: 8
Tolerancia calculada: 0 %
Tolerancia incertidumbre:10 %
Tolerancia Total:10 %
Parametro<Height>
Valor medio: 392
Tolerancia calculada: 5 %
Tolerancia incertidumbre:10 %
Tolerancia Total:15 %
Parametro<Width>
Valor medio: 666
Tolerancia calculada: 17 %
Tolerancia incertidumbre:10 %
Tolerancia Total:27 %

Ilustración 62: Caso de prueba 15



Parametro<PixelDensity>
Valor medio: 49992
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %
Parametro<HSenseChange>
Valor medio: 22
Tolerancia calculada: 8 %
Tolerancia incertidumbre:10 %
Tolerancia Total:18 %
Parametro<VSenseChange>
Valor medio: 22
Tolerancia calculada: 13 %
Tolerancia incertidumbre:10 %
Tolerancia Total:23 %
Parametro<Height>
Valor medio: 229
Tolerancia calculada: 6 %
Tolerancia incertidumbre:10 %
Tolerancia Total:16 %
Parametro<Width>
Valor medio: 665
Tolerancia calculada: 1 %
Tolerancia incertidumbre:10 %
Tolerancia Total:11 %

Ilustración 63: Caso de prueba 16

Parametro<PixelDensity>
Valor medio: 42190
Tolerancia calculada: 1 %
Tolerancia incertidumbre:10 %
Tolerancia Total:11 %
Parametro<HSenseChange>
Valor medio: 25
Tolerancia calculada: 7 %
Tolerancia incertidumbre:10 %
Tolerancia Total:17 %
Parametro<VSenseChange>
Valor medio: 23
Tolerancia calculada: 8 %
Tolerancia incertidumbre:10 %
Tolerancia Total:18 %
Parametro<Height>
Valor medio: 385
Tolerancia calculada: 11 %
Tolerancia incertidumbre:10 %
Tolerancia Total:21 %
Parametro<Width>
Valor medio: 519
Tolerancia calculada: 3 %
Tolerancia incertidumbre:10 %
Tolerancia Total:13 %

Ilustración 64: Caso de prueba 17

4.2. Diagramas

A continuación se muestran los datos en gráficas incluyendo una breve aclaración.

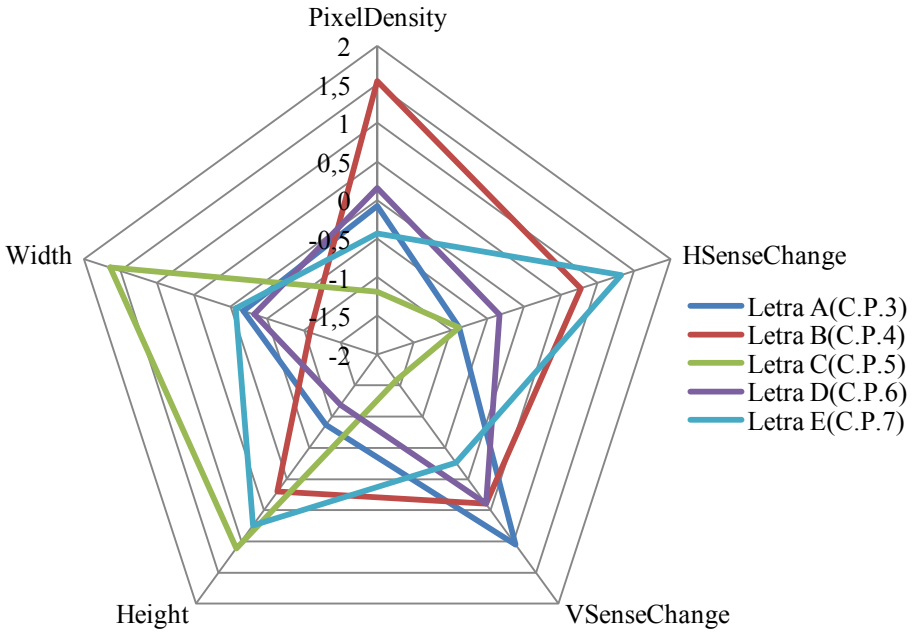


Ilustración 65: Análisis Letras

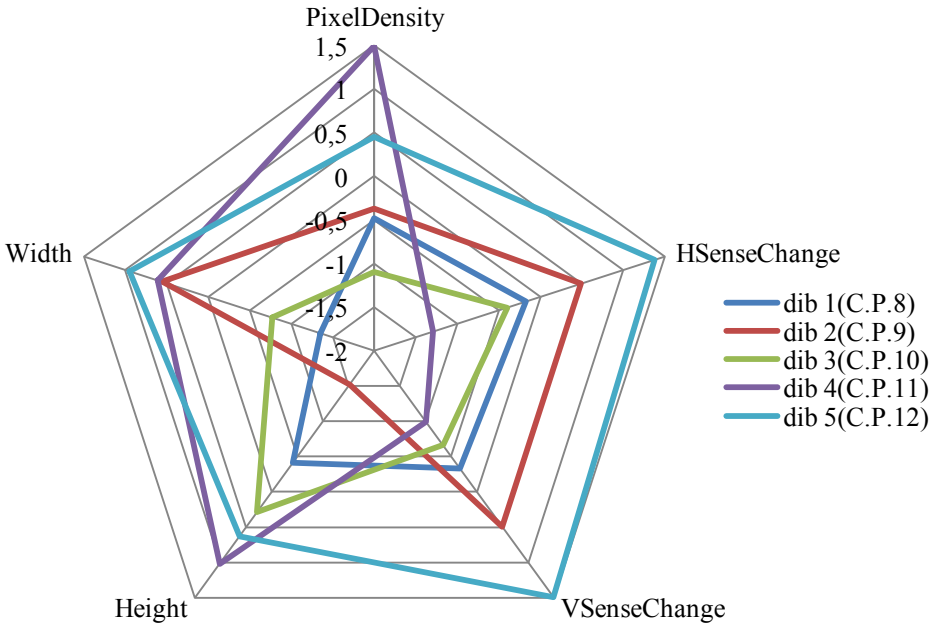


Ilustración 66: Análisis Dibujos

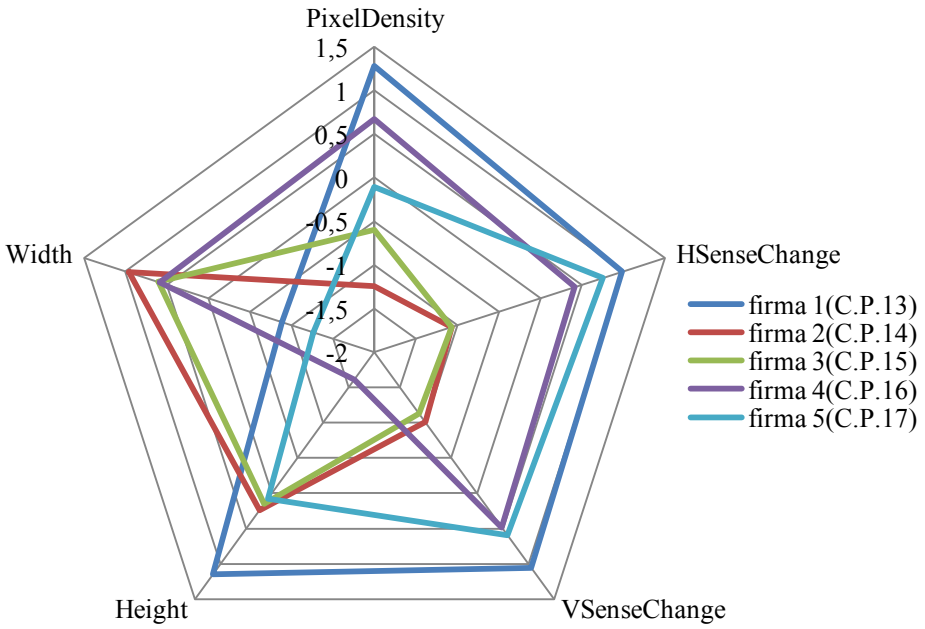


Ilustración 67: Análisis Firmas

Los datos que se han introducido en estas gráficas, han pasado un proceso de normalización. Su objetivo es analizar si ante distintas entradas, es decir trazos, obtienen unas salidas bien diferenciadas.

Es deseable que se produzca la menor colisión posible, es decir que las salidas sean lo más heterogéneas posibles. En estas gráficas se puede ver a simple vista que las salidas que proporciona el sistema son suficientemente distintas unas de otras como para poder alcanzar unos niveles de seguridad aceptables tratándose de un entorno de aplicación como el que estamos hablando.

En estos casos se han analizado 5 parámetros, pero recordemos que el número de parámetros a analizar es fácilmente ampliable, aumentando así las posibles combinaciones y aumentando de forma exponencial la seguridad del sistema, ya que si no se han producido formas similares sobre las gráficas anteriores siendo ésta un pentágono, es de esperar que si fuese un hexágono o heptágono, las posibilidades de encontrar dos trazos con los mismos valores en sus parámetros será aun menor ya que estaremos añadiendo complejidad al sistema.

En las siguientes gráficas que se van a mostrar, se ha querido representar de forma visual lo que sería de forma conceptual los cilindros de la cerradura creada. Lo interesante de las siguientes gráficas es poder comprobar que cada grupo de 5 cilindros es único para cada trazo realizado.

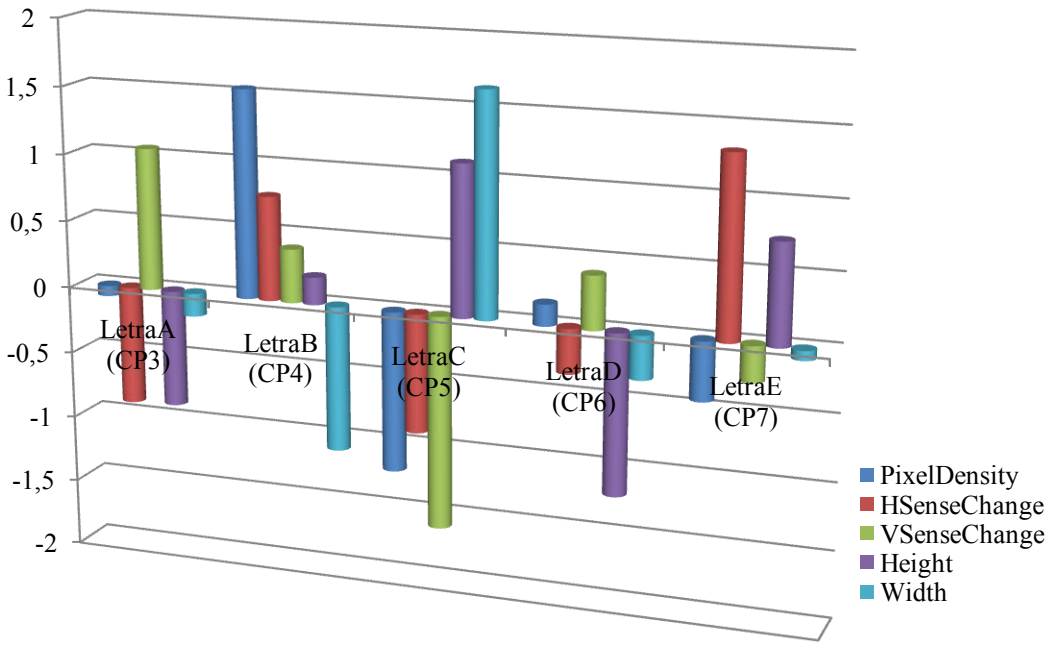


Ilustración 68: Análisis Letras 2

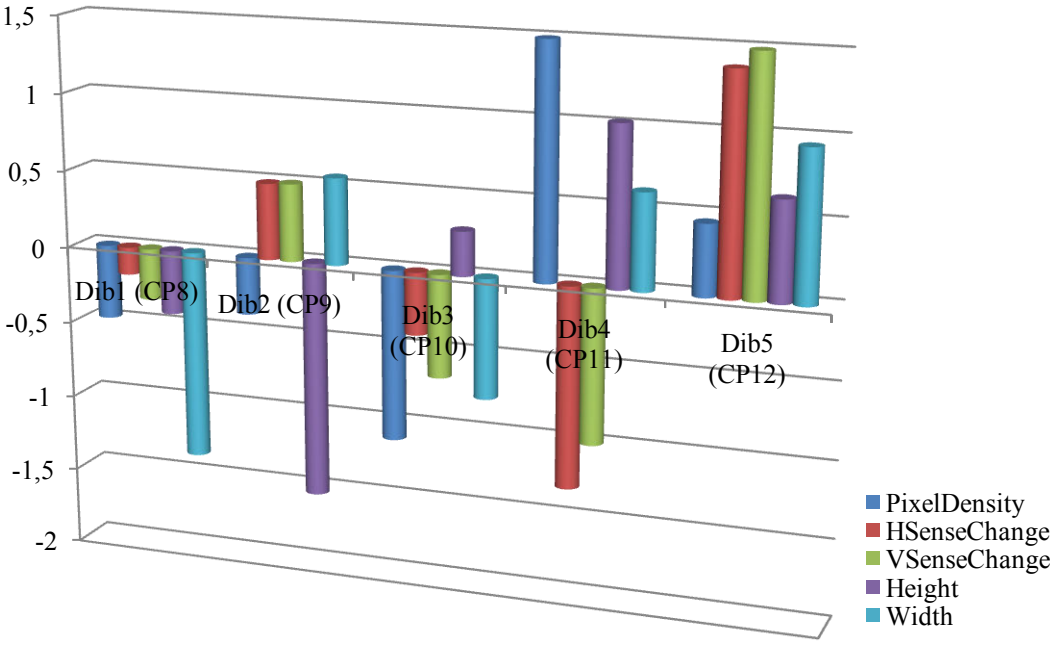


Ilustración 69: Análisis Dibujos 2

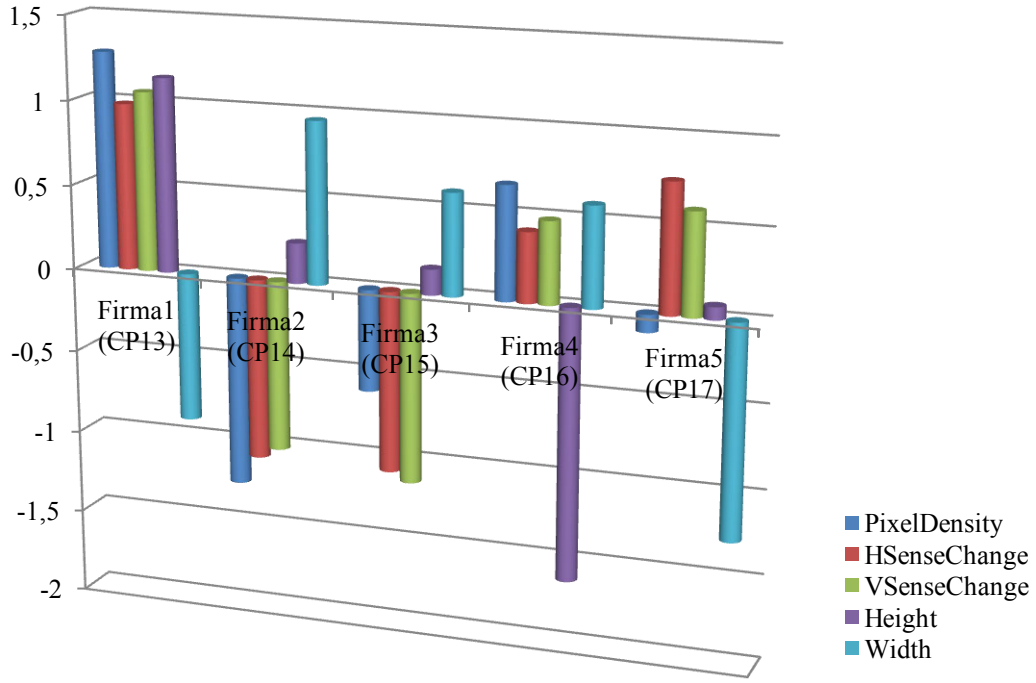


Ilustración 70: Análisis Firmas 2

Con respecto a estas gráficas comentar brevemente que casi todos los trazos realizados presentan representaciones de cilindros diferentes excepto quizá el C.P.14 y C.P.15 que presentan un patrón bastante parecido, sólo difiriendo notablemente en cuanto a su parámetro *PixelDensity*, ello podría incurrir en un falso positivo si el sistema no está configurado en su modo más restrictivo. No obstante si nos remitimos al trazo del Caso de Prueba 14(C.P.14) y al trazo del Caso de Prueba 15(C.P.15) podemos apreciar que la firma realizada es muy similar la una a la otra.

Hay que tener en consideración que en el caso de las firmas, al ser realizadas por la misma persona, se muestren patrones semejantes, es por ello que aunque se haya intentado realizar 5 firmas distintas, 2 de ellas hayan salido bastante semejantes, cosa que no se percibió durante las pruebas y que se tuvo constancia de ello al analizar los datos y observar las gráficas que muestran las propiedades de cada trazo (*Ilustración 70*).

Por ello es muy importante determinar si queremos tener un sistema muy restrictivo o algo más permisivo ya que un sistema excesivamente restrictivo eliminará la posibilidad de falsos positivos casi en su totalidad, pero puede llegar a ser frustrante no conseguir repetir exactamente el mismo trazo que registramos en el sistema.

En la siguiente gráfica se pueden observar las tolerancias de cada parámetro de todos los casos de prueba realizados para la documentación.

En la siguiente gráfica serán deseables valores de tolerancia bajos, ya que eso significa que los trazos son estables y que la fiabilidad del sistema será mayor. Recordemos que la tolerancia calculada es la máxima varianza expresada en forma de porcentaje de variación respecto de la media. Es por ello que valores bajos deben ser entendidos como trazos que siempre se realizan de forma bastante similar por parte del usuario.

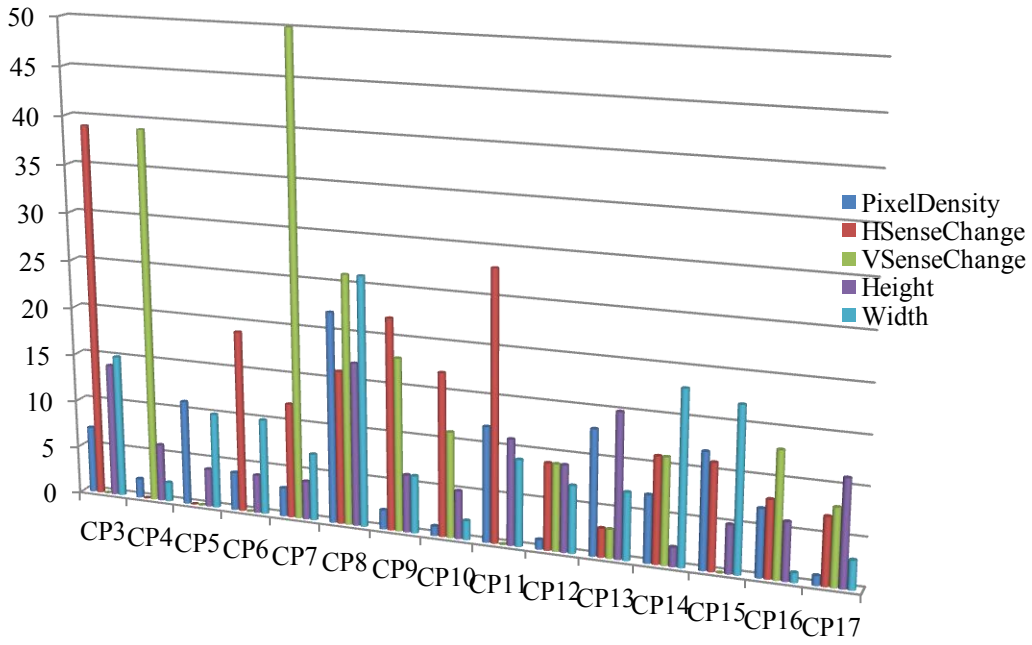


Ilustración 71: Análisis de Tolerancias

Si recordamos los grupos a los que pertenecen los casos de prueba, del CP3 al CP7 se corresponden con letras, del CP8 al CP12 se corresponden con dibujos simples y del CP13 al CP17 se corresponden con firmas.

Parece que a tenor de los valores de tolerancia mostrados, las firmas se muestran como el grupo más estable de trazos. No obstante si nos fijamos en un caso de prueba de forma individual, el comportamiento del C.P.12 es de especial interés debido a que muestra unos valores de tolerancia muy estables y moderados en todos sus parámetros analizados. En la siguiente sección *Conclusiones* se valorará cual es el comportamiento general del sistema teniendo en cuenta todos estos factores.

4.3. Tiempos de Respuesta

Iniciando fabricación de Cerradura
Iniciando quantización de PixelDensityKeyParam
Fin quantización de PixelDensityKeyParam ha tardado: 120 ms
Iniciando quantización de PixelDensityKeyParam
Fin quantización de PixelDensityKeyParam ha tardado: 60 ms
Iniciando quantización de PixelDensityKeyParam
Fin quantización de PixelDensityKeyParam ha tardado: 60 ms
Parametro<PixelDensity>
    Valor medio: 32461
    Tolerancia calculada: 3 %
    Tolerancia incertidumbre:40 %
    Tolerancia Total:43 %
Iniciando quantización de HSenseChangeKeyParameter
Fin quantización de HSenseChangeKeyParam ha tardado: 0 ms
Iniciando quantización de HSenseChangeKeyParameter
Fin quantización de HSenseChangeKeyParam ha tardado: 0 ms
Iniciando quantización de HSenseChangeKeyParameter
Fin quantización de HSenseChangeKeyParam ha tardado: 0 ms
Parametro<HSenseChange>
    Valor medio: 3
    Tolerancia calculada: 0 %
    Tolerancia incertidumbre:40 %
    Tolerancia Total:40 %
Iniciando quantización de VSenseChangeKeyParam
Fin quantización de VSenseChangeKeyParam ha tardado: 0 ms
Iniciando quantización de VSenseChangeKeyParam
Fin quantización de VSenseChangeKeyParam ha tardado: 0 ms
Iniciando quantización de VSenseChangeKeyParam
Fin quantización de VSenseChangeKeyParam ha tardado: 0 ms
Parametro<VSenseChange>
    Valor medio: 0
    Tolerancia calculada: 0 %
    Tolerancia incertidumbre:39 %
    Tolerancia Total:39 %
Iniciando quantización de HeightKeyParameter
Fin quantización de HeightKeyParameter ha tardado: 0 ms
Iniciando quantización de HeightKeyParameter
Fin quantización de HeightKeyParameter ha tardado: 0 ms
Iniciando quantización de HeightKeyParameter
Fin quantización de HeightKeyParameter ha tardado: 0 ms
Parametro<Height>
    Valor medio: 428
    Tolerancia calculada: 6 %
    Tolerancia incertidumbre:40 %
    Tolerancia Total:46 %
Iniciando quantización de WidthKeyParameter
Fin quantización de WidthKeyParameter ha tardado: 0 ms
Iniciando quantización de WidthKeyParameter
Fin quantización de WidthKeyParameter ha tardado: 0 ms
Iniciando quantización de WidthKeyParameter
Fin quantización de WidthKeyParameter ha tardado: 0 ms
Parametro<Width>
    Valor medio: 774
    Tolerancia calculada: 4 %
    Tolerancia incertidumbre:40 %
    Tolerancia Total:44 %
Fin fabricación Cerradura ha tardado: 260 ms

Como puede apreciarse los tiempos de respuesta en el análisis son muy bajos, el único valor que ha sido mayor que 0 es el análisis de la densidad de pixeles. En esta ocasión se han analizado 3 trazos repetidos en el proceso de registro que se corresponde con la fabricación de la cerradura, en un tiempo de 260ms.

Como dato a tener en cuenta, las pruebas se realizaron en un equipo con las siguientes características:

Procesador:	Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz 2.93 GHz
Memoria instalada (RAM):	8,00 GB
Tipo de sistema:	Sistema operativo de 64 bits

Es posible que en otros equipos los tiempos en la fabricación difieran respecto de los aquí presentados.

Capítulo 5: Conclusiones

5.1. Bases de Verificación y Validación

La verificación de un producto consiste en confirmar que los productos desarrollados reflejan de forma correcta los requisitos definidos para ellos, en otras palabras la verificación asegura que se ha construido de forma correcta.

En relación a los Resultados obtenido en la sección anterior, se puede comprobar cómo los casos de Prueba cubren los Requisitos que se definieron inicialmente, alcanzando todos y cada uno de los criterios de aceptación. Es por ello que el sistema se puede considerar verificado en cuanto se han cubierto con las especificaciones requeridas y registradas en el alcance del proyecto.

La validación es el proceso que garantiza que el producto que se ha desarrollado cumple con los propósitos del uso para el que fue inicialmente concebido. En otras palabras, la validación nos asegura que hemos construido lo que deseábamos.

Se han realizado pruebas con distintos individuos alcanzando en todos ellos un alto grado de satisfacción en cuanto a la respuesta general del sistema. Se ha podido comprobar que el dispositivo uDraw desarrollado para la PS3, es el que ha obtenido una experiencia de usuario más pobre, dado el corto radio de alcance en cuanto a su interfaz wireless que se ha experimentado, así como la necesidad de ejercer una presión que a juicio de todos los implicados en las pruebas, era excesiva. El problema de la sensibilidad del que adolece el uDraw, puede ser fácilmente solucionado cambiando de tableta digitalizadora. En nuestro caso la sustitución fue por una tablet Bamboo de Wacom, de un precio ligeramente superior y con capacidad para convertirse en un dispositivo inalámbrico comprando un periférico wireless.

5.2. Análisis de los resultados obtenidos

Con los resultados anteriores se hará una valoración de cuál ha sido el comportamiento general del sistema.

5.2.1. Periféricos

Con respecto a los periféricos, ya se ha comentado a lo largo de este documento que el dispositivo uDraw de PS3 adolece de una serie de carencias en cuanto a rango de alcance wireless, así como en cuanto a sensibilidad en el reconocimientos de los trazos realizados, que hacen que la experiencia de usuario fuese algo pobre y poco fluida.

Los puntos a favor de usar este dispositivo vienen del bajo coste de la unidad así como de la posibilidad de encontrarlo a día de hoy con relativa facilidad en grandes almacenes que aun dispongan de stock, condición que cambiará al estar descontinuado por parte del fabricante THQ.

Esta serie de inconvenientes, son fácilmente eludibles mediante el cambio del dispositivo por uno de prestaciones superiores y que no sea de un coste mucho mayor. Se ha encontrado en el mercadeo algunas alternativas que provienen del fabricante Wacom y que al menos una de ella se ha podido testar. Se trata de la tableta Bamboo de Wacom, que ha dado un rendimiento muy superior a cambio de un ligero incremento de precio.

En este caso el análisis marginal que relaciona "mejora" respecto a "inversión", nos permite cierto margen en cuanto a aumentar el presupuesto ya que la mejora obtenida es realmente grande.

En concreto la solución a la que se podría migrar como primera opción si se desease incrementar las capacidades del conjunto en general sería la Tablet IntuosPen S de Wacom con un P.V.P de 54,70€ y el kit wireless Wacom de P.V.P 32,40€ a fecha del presente documento.

El dispositivo de entrada basado en el reloj de Texas Instruments Chronos eZ430 no se ha podido probar finalmente a pesar de estar el sistema preparado para ello. La experiencia con el comportamiento que ha tenido el sistema frente a la tableta, nos hace pensar que el funcionamiento será más cómodo y accesible a personas con discapacidad debido a que al tratarse de un dispositivo que va alojado en la muñeca, facilitará su uso al usuario.

5.2.2. Interfaz de usuario y accesibilidad

Con relación a la interfaz de usuario y la respuesta de la misma, indicar que la experiencia de usuario fue bastante positiva, siendo los controles accesibles y simples.

Se ha tratado de minimizar al máximo el número de pulsaciones para acceder a las funcionalidades debido a que una de las características principales que debía cumplir la aplicación era la accesibilidad, objetivo que creo ha sido alcanzado con un alto grado de satisfacción.

Para acceder a las operativas principales del sistema, se ha puesto especial cuidado en el diseño de la interfaz. El resultado final es que para acceder a dichas operativas, basta con hacer un solo click en un botón de grandes dimensiones que facilita la accesibilidad al usuario. De esta manera es posible autenticarse, cambiar el trazo validador o hacer logout del sistema de forma simple, clara y cómoda.

Para la interfaz de configuración del administrador se ha tratado de agrupar las funcionalidades en una pestaña de modo que se pueda modificar de forma rápida e intuitiva los principales valores que intervienen en la aplicación.

5.2.3. Resultados de pruebas operativas

La respuesta general del sistema es bastante buena, el análisis de los parámetros que se han definido es suficiente para proporcionar la seguridad necesaria a un entorno como el que estamos hablando.

El sistema ha sido concebido para estar dentro del hogar, por lo que las necesidades de autenticación, son menos exigentes que si se tratase de un sistema de control de acceso. Más bien el sistema está diseñado para que los distintos actuadores presentes en el HDA entren en funcionamiento o no si la persona está registrada en el sistema.

Aplicado a la seguridad en el control de acceso, el sistema podría utilizarse para activar una alarma acústica o de aviso a las autoridades si no se logra introducir el trazo correctamente en un tiempo limitado. Por ejemplo, si pasados 5 minutos desde que la puerta se ha abierto, el usuario no consigue autenticarse con la herramienta, se podría tomar algún tipo de medida que garantizase la seguridad como activar una alarma acústica o avisar a las autoridades.

Con respecto a la configuración del sistema, se ha comprobado que cuando el trazo que va a usarse como clave, se repite 3 veces o más antes de registrarlo, aumenta de forma significativa la precisión del sistema. Cuantas más veces se repita, mejor es la respuesta del sistema, si bien es cierto que se desecharon valores superiores a 5 por ser poco práctico y tedioso. A nuestro criterio, el trazo a registrar debe repetirse entre 2 y 5 veces para que el sistema tome valores medios y tolerancias a ligeras variaciones.

Con respecto al trazo para validarse en el sistema, ocurre lo mismo, cuantas más veces se repita, mejor será el rendimiento del sistema, no obstante en este caso, se ha considerado que a partir de 2 veces, empieza a ser incómodo para el usuario, ya que es una operación que se realiza de forma frecuente y a pesar de ganar en precisión, se ha preferido que primara la usabilidad y rapidez para contar con mayor aceptación.

Como dato adicional, comentar que el sistema se entregará configurado para validarse dibujando una sola vez el trazo y registrar un trazo repitiéndolo 3 veces.

Con respecto a los tipos de trazo que se han agrupado como Letras, Dibujo simples y Firmas en este apartado se comentarán algunas cosas de interés.

i. Trazos categoría Letras

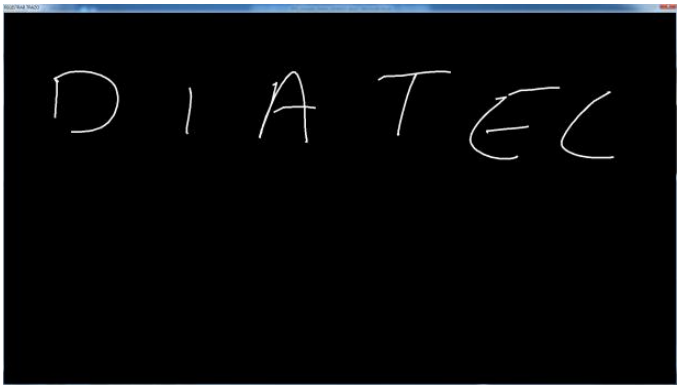
Las letras que se han analizado en los casos de prueba, han sido letras simples, mayúsculas e individuales. Se ha podido comprobar que el análisis de los parámetros arrojaba valores identificables permitiendo diferenciar una de otra, sin embargo, los trazos tan simples como una letra, tienen un particular problema y es que cualquier ligera modificación, supone que la tolerancia calculada se dispare.

Por ejemplo la letra C con sólo 1 cambio de sentido horizontal, hace que si en uno de los trazos a repetir para registrarla, nos sale un ligero arrastre de lápiz creando un segundo cambio de sentido, la tolerancia se dispare al 50% valor tremendamente alto que hará el sistema más vulnerable.

El problema es que aparte de ser un valor alto, es inestable, ya que no siempre se produce y es imprevisible lo cual es un problema añadido de cara a gestionarlo o compensarlo con la tolerancia de incertidumbre.

Es por ello que las letras o trazos extremadamente sencillos, introducen demasiada holgura al sistema y deben ser evitados.

Se sugiere que se combinen entre ellas para añadir cierta complejidad y reducir el fenómeno no deseable de la holgura. Con una combinación como la siguiente se estabilizan los niveles de tolerancia manteniéndolos en valores perfectamente admisibles.



ii. Trazos categoría Dibujos simples

Los dibujos simples son bastante heterogéneos en cuanto a los valores que arrojan en su análisis de parámetros como puede apreciarse en la ilustración 66. Son un buen trazo a la hora de generan combinaciones posibles, ya que puede ser cualquier cosa que el usuario decida.

El mayor problema que presentan, es que hay algunos dibujos, en concreto los que no están basados en formas geométricas, que muestran cierta tendencia a tolerancias estables en todos sus parámetros, pero algo elevadas todavía.

Si bien es cierto que al ser conscientes de que este tipo de trazo introduce ya unos valores de tolerancia medios, podemos compensar bajando el nivel de tolerancia de incertidumbre que es configurable por el usuario a través de la interfaz.

Dentro de los trazos de la categoría Dibujos simples los que han ofrecido mejores resultados son los que se corresponden con la combinación de varias formas geométricas simples en un mismo trazo. C.P.12 que muestra unos niveles de tolerancia bajos y estables

iii. Trazos categoría Firmas

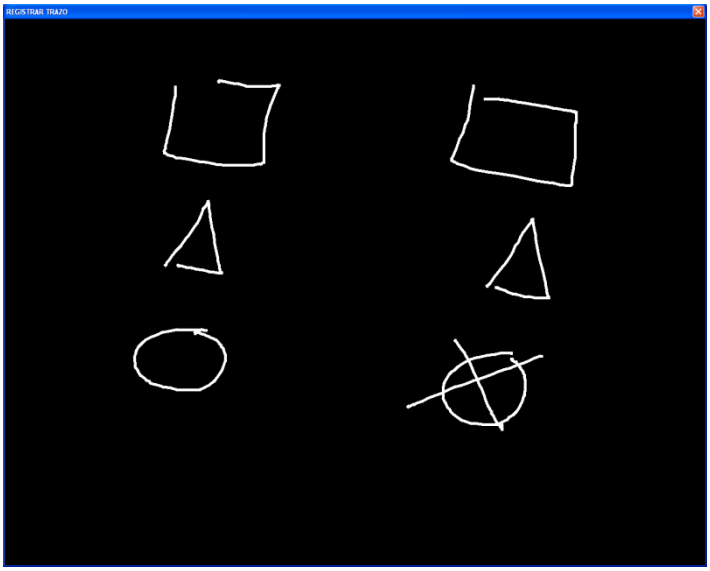
Las firmas han tenido un comportamiento muy bueno en el sistema. Son variadas, sobre todo si las realizan personas distintas, y son bastante estables en cuanto a sus valores de tolerancia, debido a que estamos muy acostumbrados a realizarlas para multitud de trámites.

Uno de los problemas principales que se han encontrado es que en particular, 2 de los parámetros, el relativo a la altura y la anchura, son muy similares en todos los casos de prueba. Ello podría ser debido a que normalmente la firma la realizamos en espacios destinados a ella con dimensiones restringidas y la mayoría de la gente firma siguiendo un movimiento que responde al juego de muñeca, con lo que el área máxima que abarca está limitada y con ello los parámetros de anchura y altura.

Otro de los problemas es que la firma suele ser conocida o accesible por gente de nuestro entorno, con lo que se convierte en un elemento de seguridad relativamente expuesto.

iv. Mejor tipo de trazo

Si finalmente nos tuviésemos que decantar por un tipo de trazo frente a los demás, a tenor de los resultados obtenido en las pruebas se puede interpretar que la combinación de figuras geométricas variadas son la mejor opción en cuanto a variabilidad de sus parámetros analizados, así como en los valores de tolerancia que ofrecen. Un ejemplo de este tipo de trazo sería el siguiente:



Los resultados muestran mucha estabilidad en la forma que tenemos de ejecutar de forma repetitiva figuras geométricas básicas, que a su vez adquieren complejidad cuando combinamos varias de ellas entre sí. Es el efecto de agregación de figuras simples lo que convierte esta opción como la mejor de acuerdo a las pruebas realizadas.

Que a su vez podría combinarse con cualquiera de las otras categorías.

5.2.4. Comentarios finales

En el transcurso de este proyecto, se ha trabajado mucho en la parte previa al desarrollo. Se han buscado modelos similares y analizado características grafológicas que pudieran ayudar en el modelado de parámetros a analizar El análisis que precede a la implementación así como la analogía con el sistema real de la Llave-Cerradura han sido claves para que finalmente el sistema haya respondido de forma satisfactoria.

El sistema es fácilmente ampliable mediante la adición de nuevos parámetros, así como sucede en el sistema real que mejoran la seguridad aumentando la complejidad de las llaves y cerraduras sin que la filosofía general cambie. En este caso la seguridad aumenta con la adición de nuevos parámetros a analizar mediante una simple modificación en el fichero de configuración y el uso de introspección, lo que hace que el sistema sea flexible y escalable.

Por otra parte comentar que ha sido muy gratificante ver cómo tras tratar de modelar los distintos elementos y extraer la esencia que hacía que el sistema original funcionase, el proyecto ha ido cogiendo forma cuando se virtualizaban conceptos. La primera prueba de concepto que se realizó fue ya un éxito, siendo el sistema capaz de discriminar distintos trazos con un nivel de precisión bastante alto.

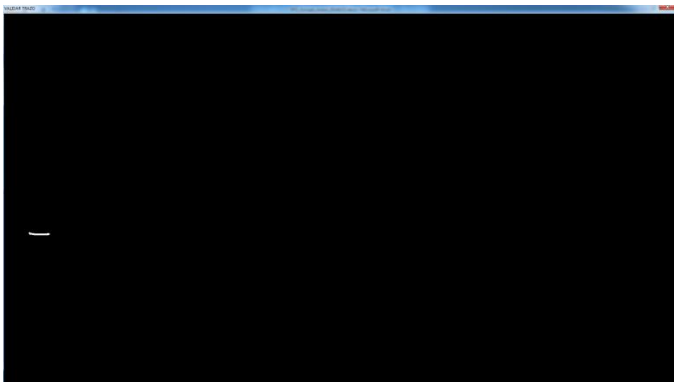
La modificación más importante que se hizo respecto del diseño original a raíz de los resultados de las pruebas, fue la inclusión de una tolerancia de incertidumbre que inicialmente no estaba contemplada. Su inclusión fue necesaria porque se pudo apreciar que el sistema era demasiado exigente al aplicar la comprobación respecto de los valores calculados, lo que en ocasiones causaba cierta frustración al no ser uno mismo capaz de repetir el trazo realizado previamente.

Por ello se decidió parametrizar esta tolerancia y dejarla a elección del usuario y sus preferencias. Se le llamo tolerancia de incertidumbre porque efectivamente no responde a otro criterio que la precisión del usuario y su capacidad para realizar el mismo trazo de forma idéntica.

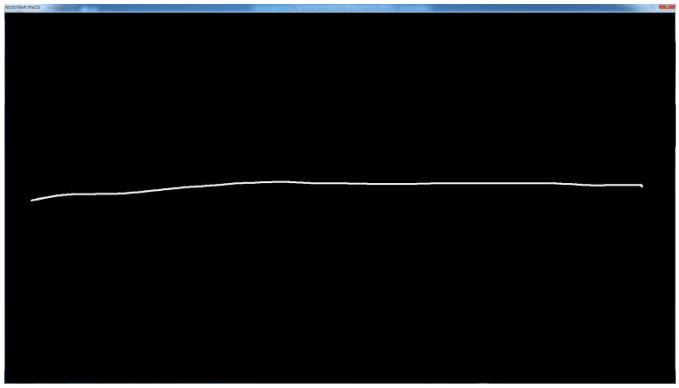
La experiencia con el sistema, nos dice que este parámetro depende tanto del usuario como del dispositivo utilizado, ya que el grado de precisión que el usuario es capaz de alcanzar, depende mucho de la calidad del dispositivo, siendo posible alcanzar mayor precisión con la tableta de Wacom que con el uDraw y por permitir un nivel de tolerancia de incertidumbre menor lo que aumenta la seguridad al hacer el sistema más preciso.

La aplicación basa su seguridad en el secreto de la forma elegida por el usuario, que puede ser desde una simple línea, un dibujo o su firma. La aplicación no basa su seguridad en discriminar si un círculo lo ha realizado el usuario A o el usuario B, aunque también influye por los parámetros analizados, no es el propósito de la herramienta. La aplicación basa su seguridad en que el usuario A y sólo él sabe que su trazo es un círculo, o un círculo con un cuadrado en su interior, o su firma, o 3 letras. En este caso el espacio muestral es infinito, ya que no hay limitación en cuanto a las posibles combinaciones que se permiten, no así como ocurre con un PIN de 4 dígitos en el que las posibles combinaciones son sólo 10.000.

En este caso, por poner un ejemplo, una simple línea recta, dependiendo de su longitud puede tener un valor de 882 como es el caso:



O un valor de 26331 como en el siguiente caso:



Lo que hace que sólo analizando un parámetro ya tengamos un rango de valores mayor que con el PIN convencional. Siempre y cuando el trazo permanezca en secreto claro está.

Es importante fijar este concepto ya que de no ser así estaremos limitando las capacidades en cuanto a combinaciones posibles que otorgan mayor seguridad al sistema.

Una vez revelado el secreto del trazo escogido por el usuario, la seguridad se ve comprometida, en este caso su comportamiento es algo mejor que lo que sucede con un PIN, ya que si conocemos el PIN de la tarjeta de crédito del usuario A, la seguridad está completamente rota, sin embargo, en este caso, aunque sepamos que el usuario A tiene como trazo su firma, y tengamos su firma delante, es probable que si intentamos reproducirla, haya ligeras variaciones que el usuario A haga a la hora de escribir, que haga que no consigamos reproducirla en los primeros intentos, no obstante, la seguridad se habrá visto seriamente comprometida ya que será cuestión de tiempo que consigamos reproducir su firma de tal manera que el sistema la acepte como válida.

Es por ello que considero muy importante hacer especial hincapié en que el trazo que decida utilizar el usuario debe ser secreto como si de un PIN se tratase.

5.3. Futuras líneas de trabajo

A continuación se nombrarán algunas de las posibles mejoras que podrían efectuarse sobre el sistema.

5.3.1. Multiusuario

Se podría dar capacidad para soporte multiusuario, pudiendo asignar cada usuario a una tecla de función de tal manera que el sistema fuese capaz de mantener varios tokens activos al mismo tiempo.

Las modificaciones a realizar no serían muy complejas ya que la estructura del fichero así como los parseadores lo permiten.

5.3.2. Distribución certificados

Sería interesante implementar un sistema para la distribución del certificado público de la aplicación de cara a comprobar la firma del token. Actualmente esta distribución ha de realizarse de forma manual.

5.3.3. Kerberos

Se podría añadir un sistema Kerberos, como infraestructura de seguridad para la gestión de las claves secretas que intervienen en la herramienta.

5.3.4. Cifrado ficheros Configuración

Para evitar la manipulación externa de los datos contenidos en ellos, sería interesante cifrar estos ficheros o bien emplear el sistema de permisos del sistema operativo y limitar el acceso a ellos permitiendo únicamente a la aplicación leer o escribir *chmod 700* y convertir a la aplicación en usuario propietario *owner* de estos ficheros.

Capítulo 6: Referencias

1. www.casadomo.com. [En línea] [Citado el: 7 de 12 de 2013.] http://www.casadomo.com/noticiasDetalle.aspx?c=9&idm=15.

2. www.domoprac.com. [En línea] [Citado el: 8 de Diciembre de 2013.] http://www.domoprac.com/protocolos-de-comunicacion-y-sistemas-domoticos/historia-de-la-domotica-pasado-presente-y-futuro/1.-introduccion-la-revolucion-domotica.html.

3. www.knx.org. [En línea] [Citado el: 8 de Diciembre de 2013.] http://www.knx.org/es/knx/que-es-knx/.

4. www.knx.org. [En línea] [Citado el: 8 de Diciembre de 2013.] http://www.knx.org/fileadmin/downloads/07 - News & Press/03 - Press Releases/04 - Espagnol/PR20061201-ES.pdf.

5. www.lonmark.org. [En línea] [Citado el: 8 de Diciembre de 2013.] http://www.lonmark.org/about/.

6. www.ine.es. [En línea] [Citado el: 8 de Diciembre de 2013.] http://www.ine.es/prodyser/pubweb/discapa/disctodo.pdf .

7. www.ine.es. [En línea] [Citado el: 9 de Diciembre de 2013.] http://www.ine.es/prensa/np524.pdf.

8. www.proyectosfundacionorange.es. [En línea] [Citado el: 9 de Diciembre de 2013.] http://www.proyectosfundacionorange.es/docs/eEspana_2013_web.pdf.

9. www.euitt.upm.es. [En línea] [Citado el: 10 de Diciembre de 2013.] http://www.euitt.upm.es/estaticos/catedra-coitt/web_socioeconomica/IIIencuentroTELECO_DISCAP/ponenciasPDF+paginaWEB/17.pdf.

10. www.ibm.com. [En línea] [Citado el: 13 de Enero de 2014.] http://www.ibm.com/developerworks/webservices/library/ws-soa-term2/.

11. www.rediris.es. [En línea] [Citado el: 13 de Enero de 2014.] http://www.rediris.es/cert/doc/unixsec/node14.html.

12. www.thq.com. [En línea] [Citado el: 21 de Enero de 2014.] http://www.thq.com/us/udraw-gametablet/udraw_playstation®3.

13. www.ti.com. [En línea] [Citado el: 21 de Enero de 2014.] https://estore.ti.com/eZ430-Chronos-433-Chronos-Wireless-development-tool-in-a-watch-P1734.aspx.

14. brandonw.net. [En línea] [Citado el: 21 de Enero de 2014.] http://brandonw.net/udraw/ .

15. www.wacomonline.com. [En línea] [Citado el: 21 de Enero de 2014.] http://www.wacomonline.com/c/15/INTUOS/.

16. xiph.org. [En línea] [Citado el: 24 de enero de 2014.] http://xiph.org/~xiphmont/demo/daala/demo1.shtml.

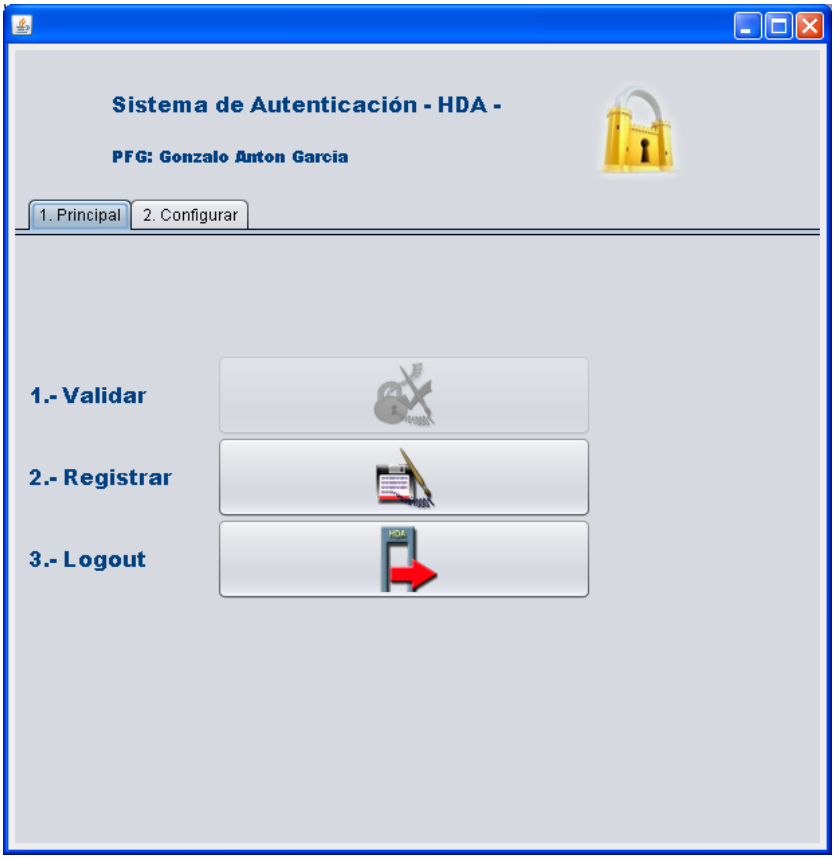
17. Chrissis, Mary Beth, Konrad, Mike y Shrum, Sandy. *CMMI for development, version 1.2*. s.l. : Addison Wesley, 2006. ISBN/0-321-27967-0.

Anexos

A. Manual de Usuario

A.1. Primeros Pasos

Al iniciar la aplicación por primera vez, el usuario encontrará la siguiente pantalla.



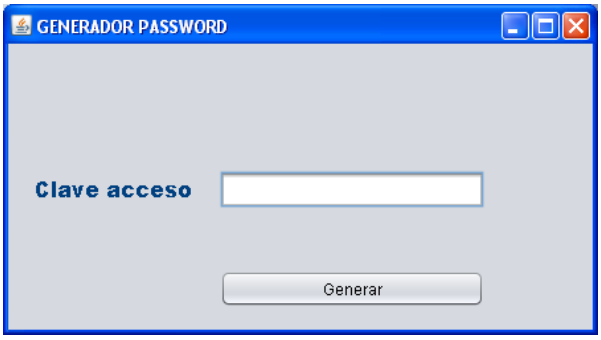
En ella no es posible comenzar con el proceso de validación ya que el sistema aún no cuenta con un Trazo previamente registrado.

Para ello el Administrador deberá proceder en primer lugar a Configurar el sistema seleccionando la pestaña *2. Configurar*

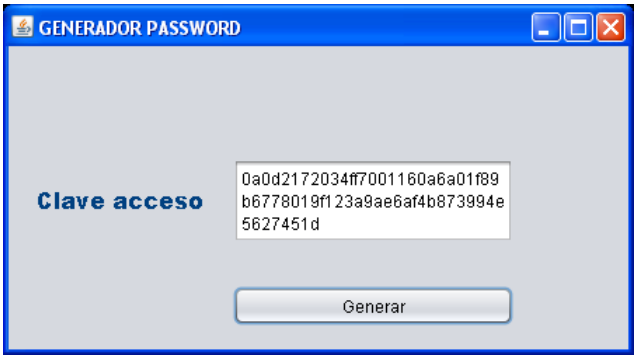
En la siguiente pantalla se nos solicitará que introduzcamos una contraseña.

La contraseña inicial está basada en un algoritmo secreto que toma unos determinados Identificadores únicos propios de cada máquina, como el número de serie de la BIOS, numero serie Placa Base, etc y genera una clave única para ese equipo.

Para generar esa clave se ha desarrollado como anexo a este proyecto una herramienta contenida en el directorio PFG_EXTRAS que se ejecutará al hacer doble click sobre el archivo *ejecuta_generador.bat* tal y como se muestra:



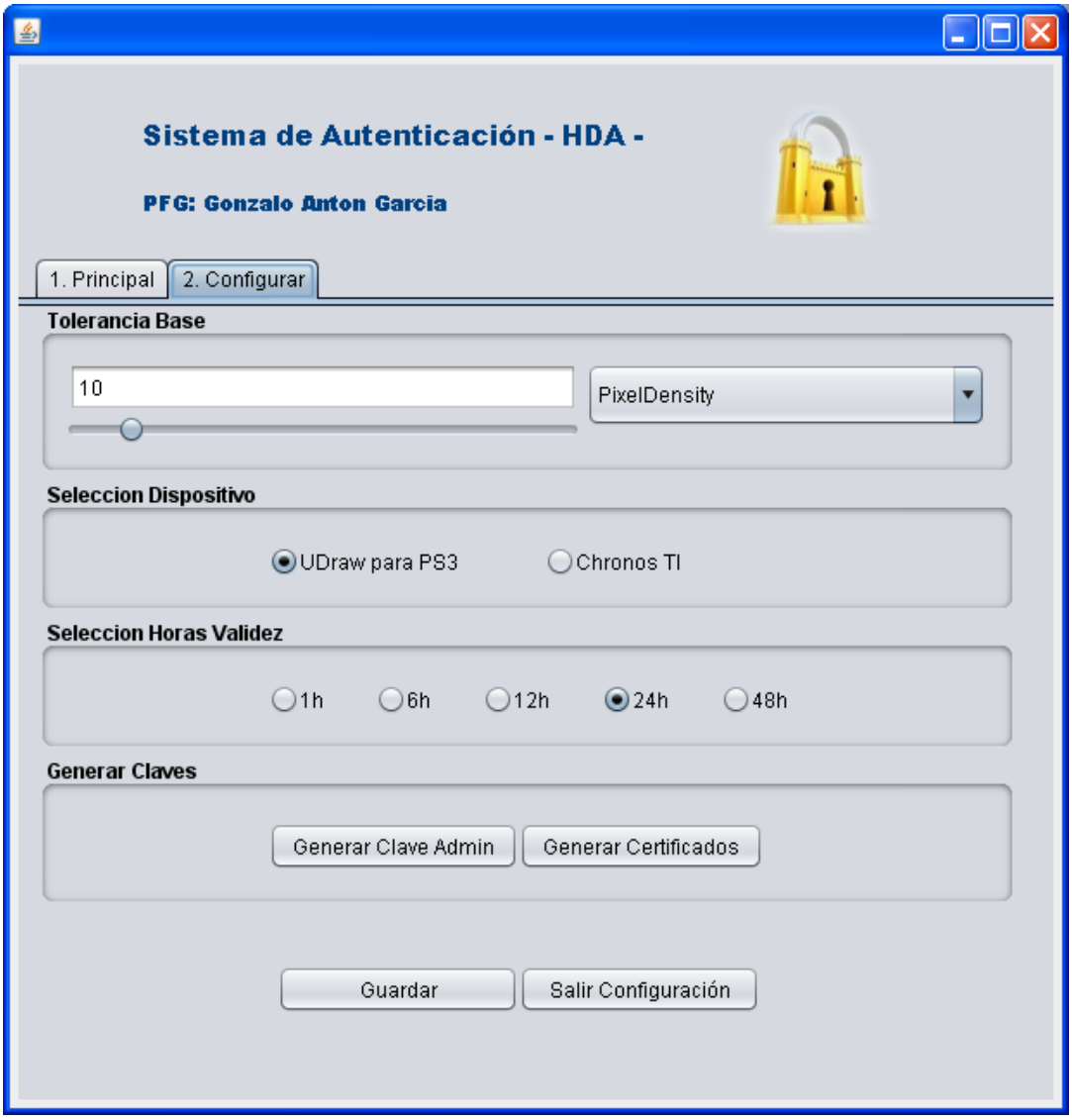
Pulsando sobre la opción *Generar* obtenemos la clave que deberemos copiar en nuestra aplicación



Copiamos la clave en la pantalla de entrada al menú de configuración y pulsamos *Entrar*.

Una vez comprobada la contraseña inicial, se nos mostrará las operaciones que el Administrador tiene permisos para configurar.

En la siguiente pantalla se nos mostrarán las distintas opciones que se pueden configurar.



La selección del dispositivo, determinará el periférico empleado para introducir el trazo.

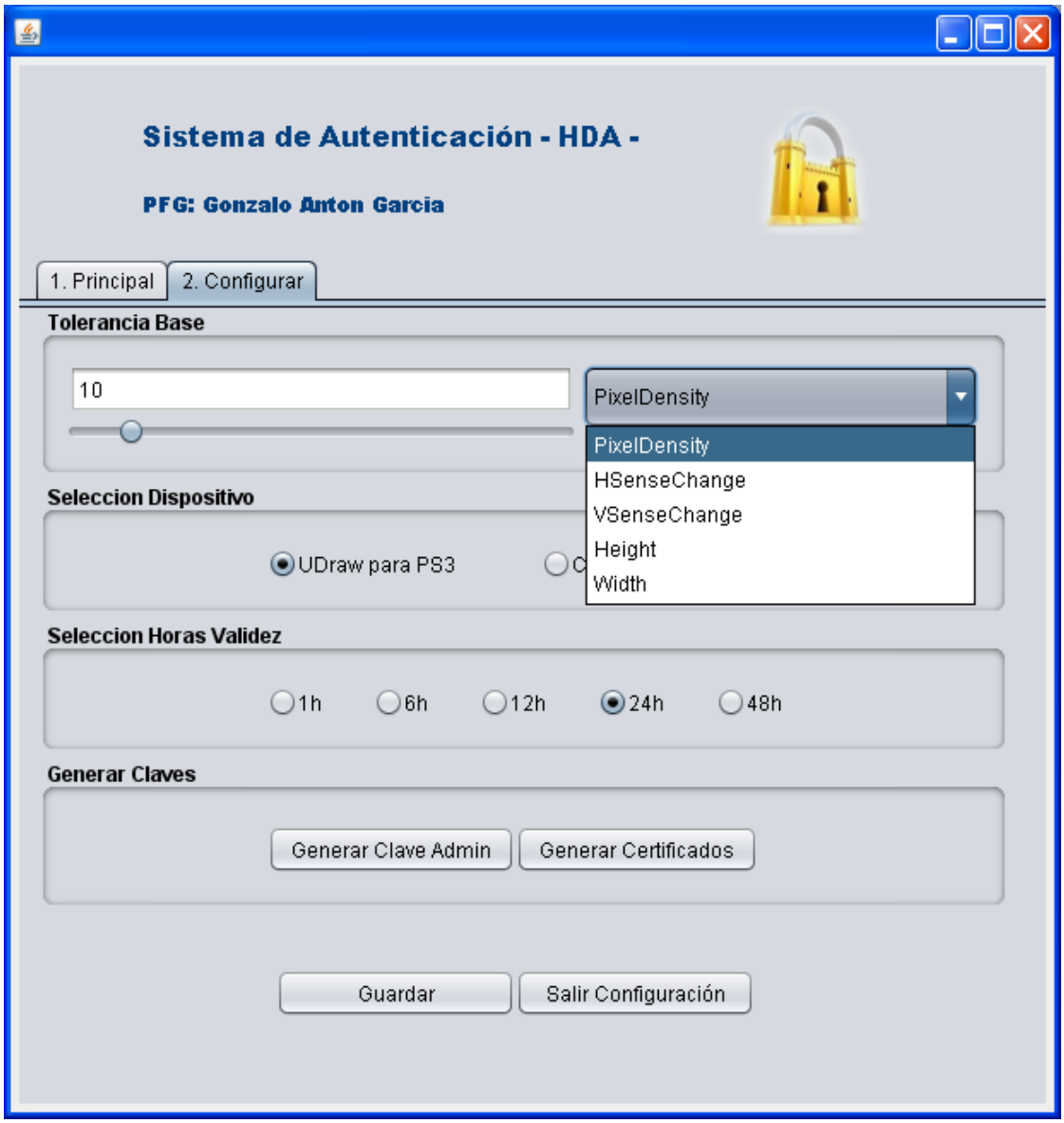
Las horas de validez, determinarán el periodo para el cual el token generado es válido, estableciendo su validez en NoAntes(fecha actual) y NoDespues(fecha actual + validez)

La generación de claves, en concreto la de certificados, es necesaria para el correcto funcionamiento de la aplicación. Se deben generar la primera vez que se use el sistema ya que la clave privada del certificado irá cifrada con una clave calculada explícitamente para la máquina en la que se ejecute.

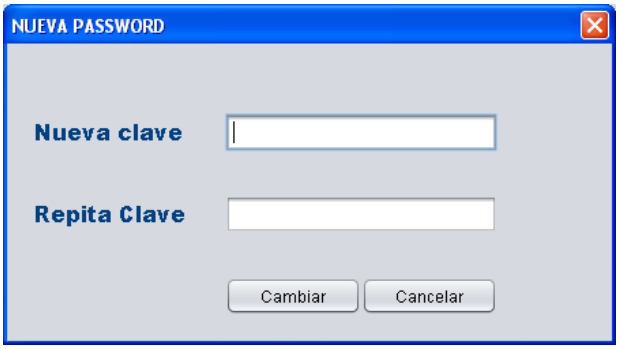
La configuración de la *Tolerancia Base*, es independiente para cada parámetro presente en el análisis del trazo, y es responsable de establecer cuánto de restrictiva va a ser esa propiedad con respecto al valor original calculado.

Un valor muy bajo hará que ese parámetro no tolere diferencias respecto al trazo previamente almacenado, haciendo muy difícil obtener la autenticación por parte del sistema.

Un valor alto hará que el sistema sea muy tolerante a variaciones con respecto al trazo almacenado y provoque falsos positivos.

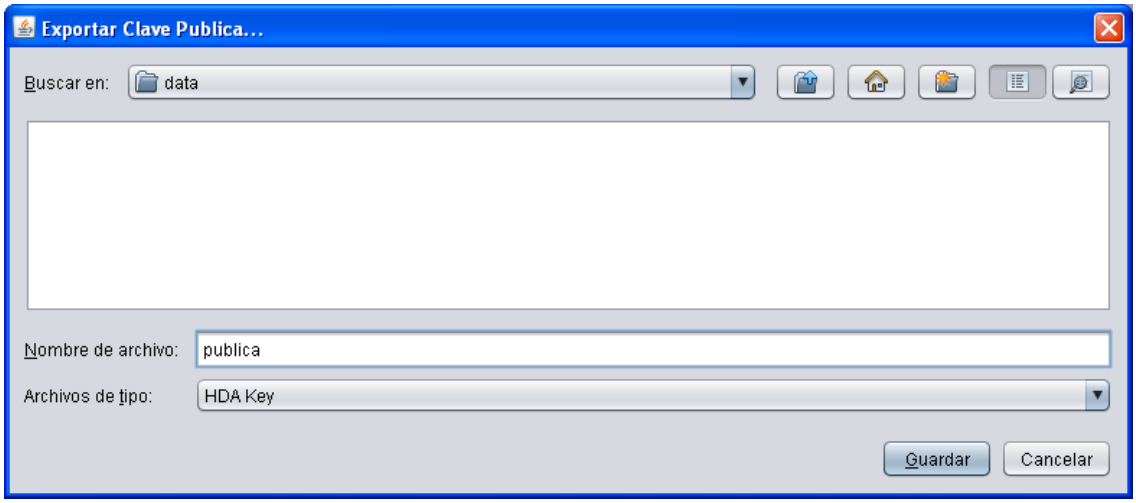


Cambiar la clave Inicial es altamente recomendable para no tener que hacer un constante uso de la herramienta de generación de password

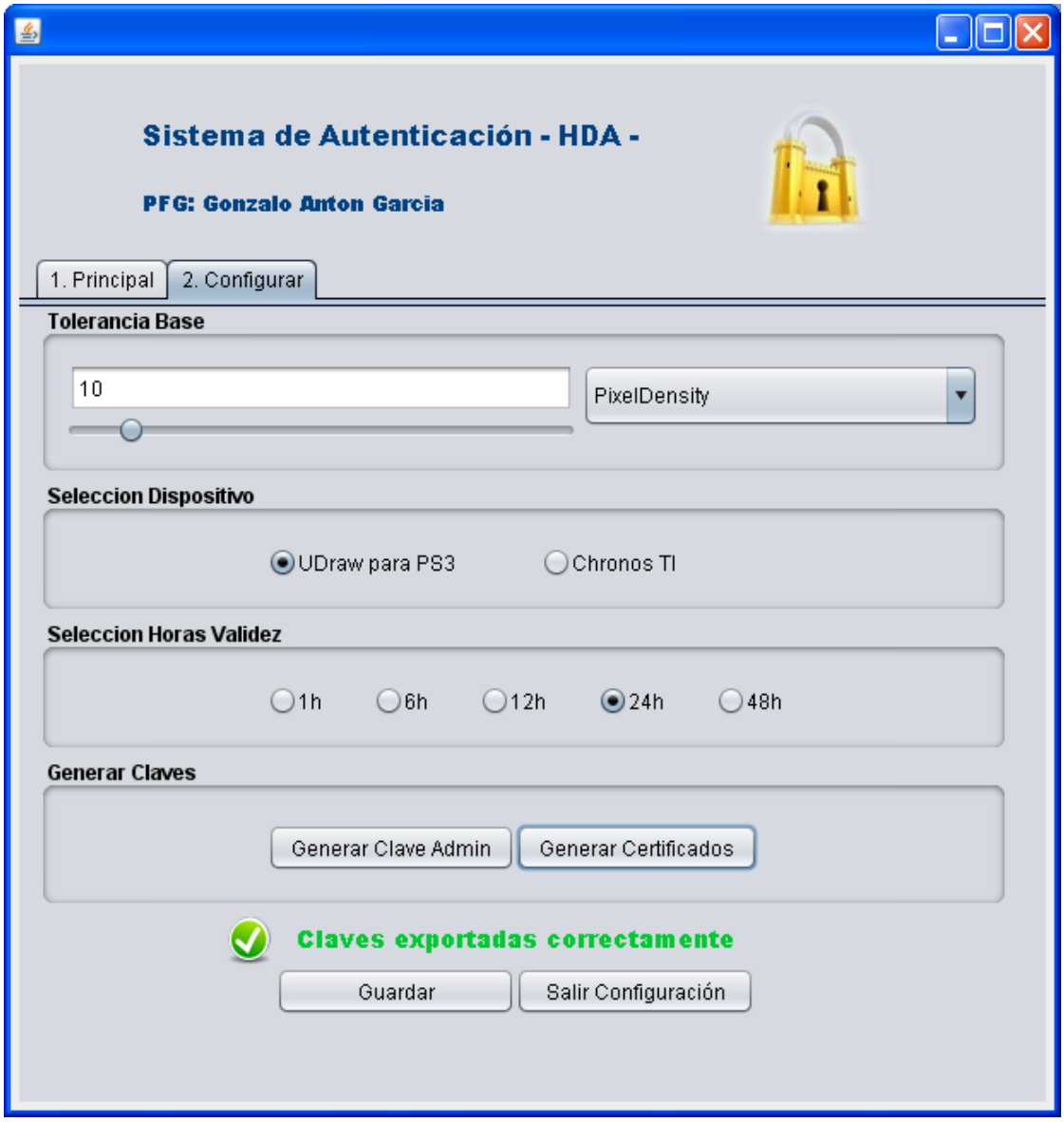


Introducimos la misma contraseña dos veces y pulsamos sobre *Cambiar*.

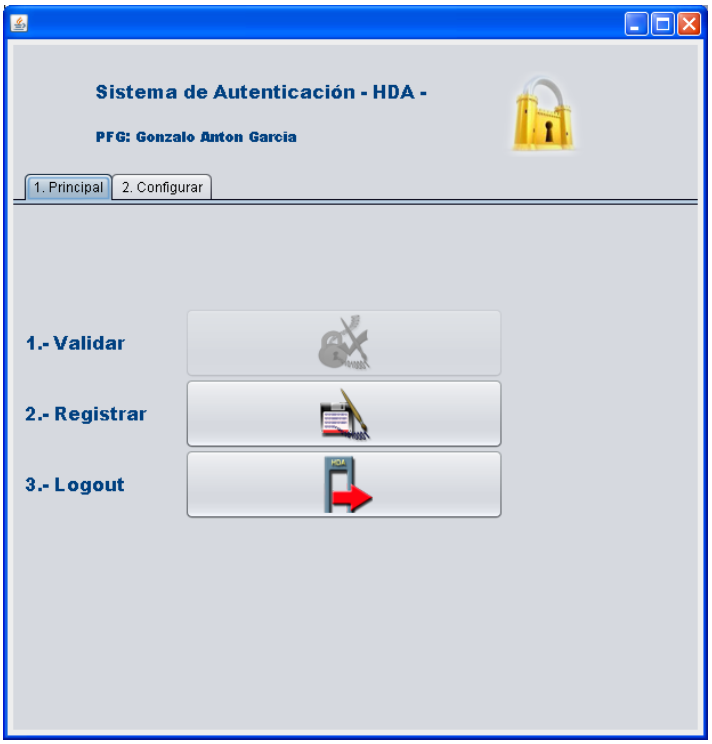
A continuación pulsamos sobre *Generar Certificados* y escogemos un nombre y directorio donde guardar la clave pública que deberemos compartir con el resto de equipos que quieran validar el token que se pretende generar.



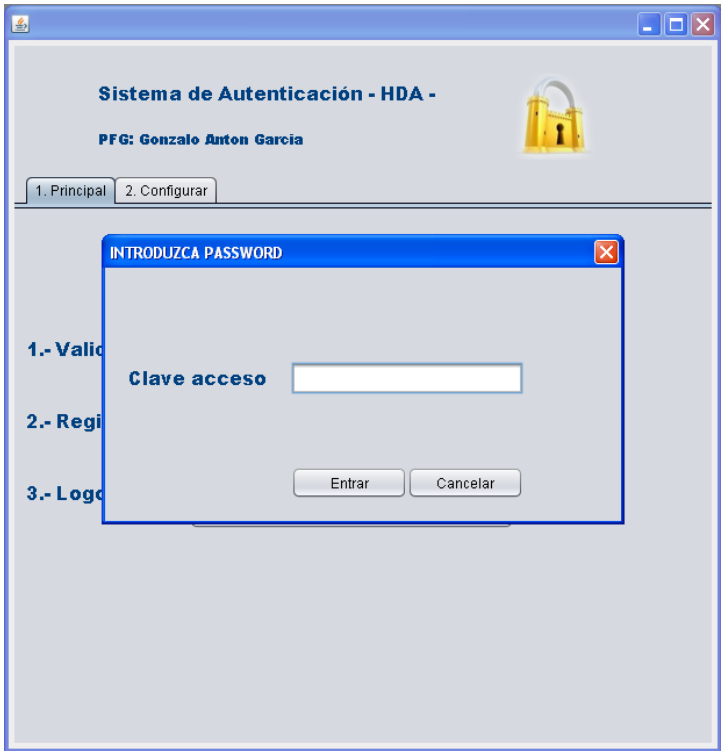
Una vez concluido el proceso de configuración, pulsamos sobre *Guardar* para salvar los cambios y *Salir Configuración* para cerrar el acceso al menú.



Salimos de la pestaña *2.Configurar* y en la pestaña principal pulsamos sobre *Registrar*.

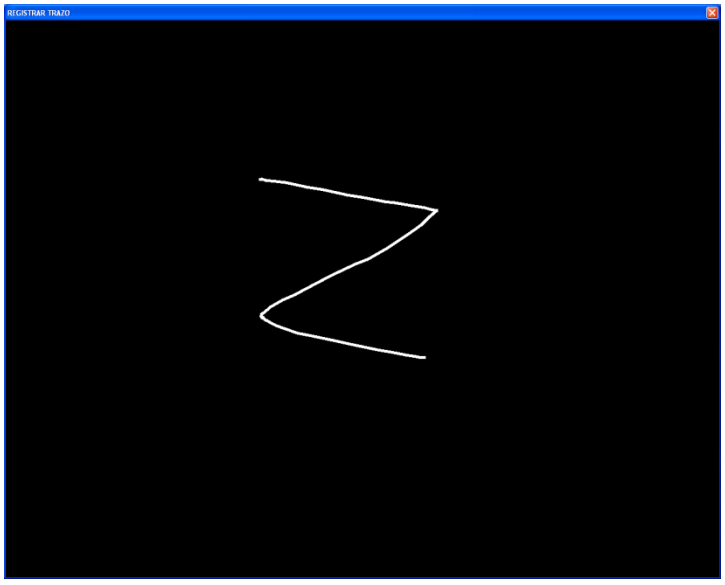


Al ser la primera vez que Registramos el trazo que servirá para Autenticarnos, nos solicita la contraseña que acabamos de establecer.



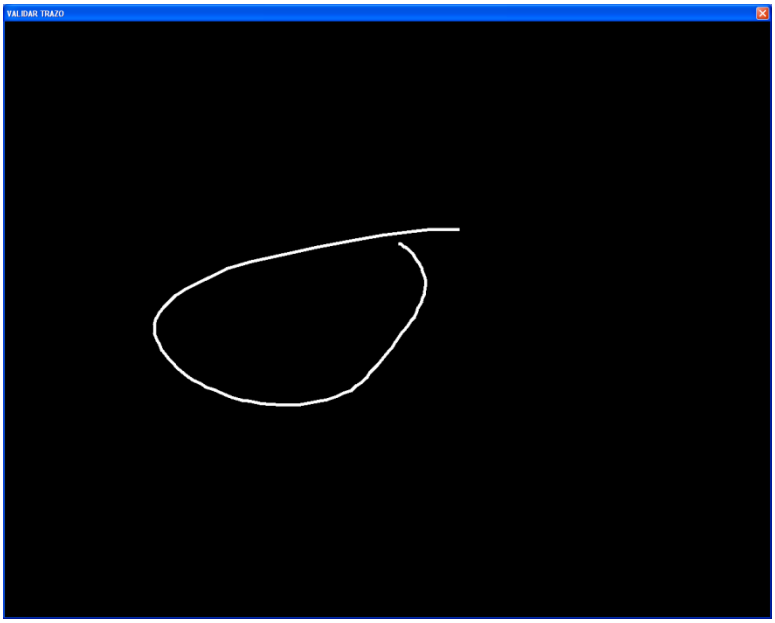
La introducimos y procedemos a realizar el trazo con el que posteriormente nos autenticaremos. En este caso la letra Z. Deberemos repetir el trazo 3 veces intentando

ser lo más precisos que podamos. El número de veces que se solicita el trazo se puede configurar en el fichero *config/lockfactoryconfig.xml* en la propiedad *numeroTrazos*



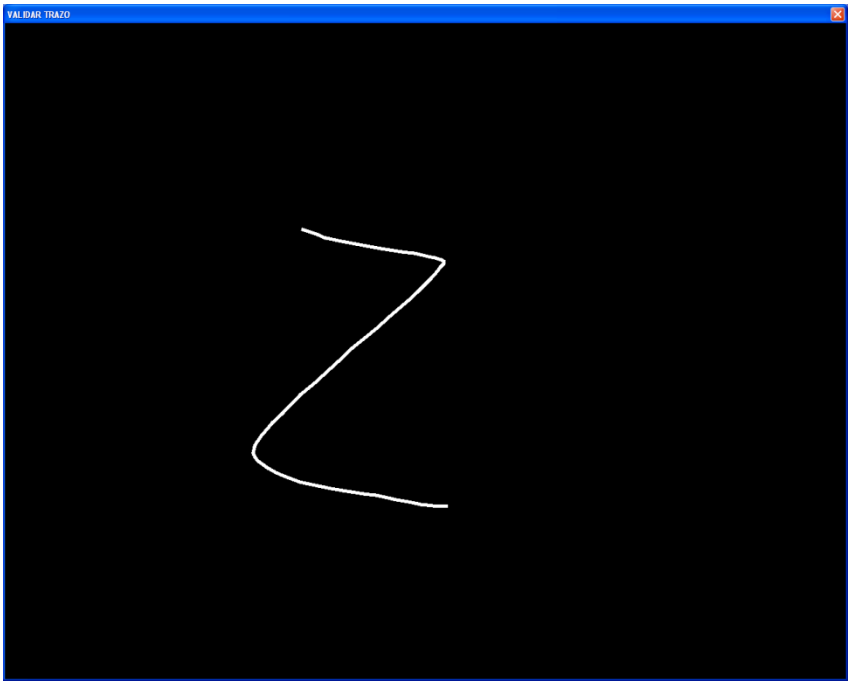
Un icono junto al botón Registrar nos indicará si el proceso ha finalizado correctamente.

Si pulsamos sobre Validar, sobre el botón central del ratón en cualquier zona o pulsamos la tecla triangular del uDraw sobre cualquier zona, podremos acceder a la ventana de validación de trazo. Efectuamos un trazo distinto para verificar que discrimina distintos trazos.



Y comprobamos que recibimos una respuesta negativa al Login.

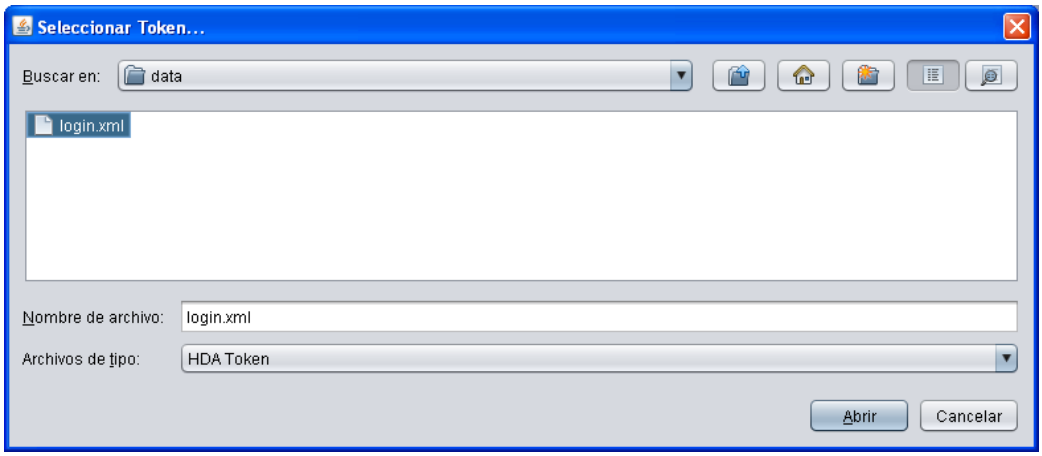
A continuación volvemos Validar, esta vez realizando el trazo que registramos previamente



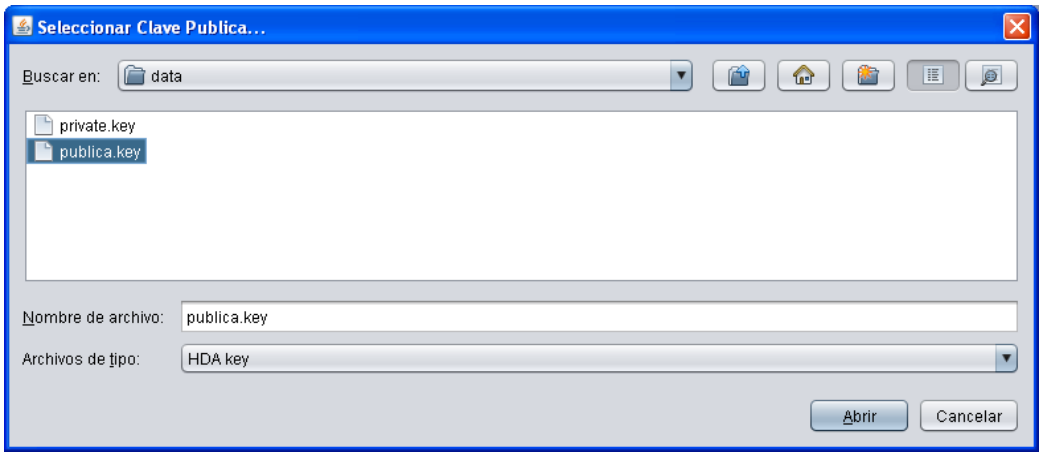
Y verificamos que el proceso ha sido realizado con éxito

Para comprobar el correcto funcionamiento del token generado, así como el funcionamiento del proceso de firmado con los certificados de la PKI, contamos con una herramienta creada como anexo en el directorio PFG_EXTRAS que se lanza a través del fichero *ejecuta_checkToken.bat.*

Seleccionaremos el token generado login.xml

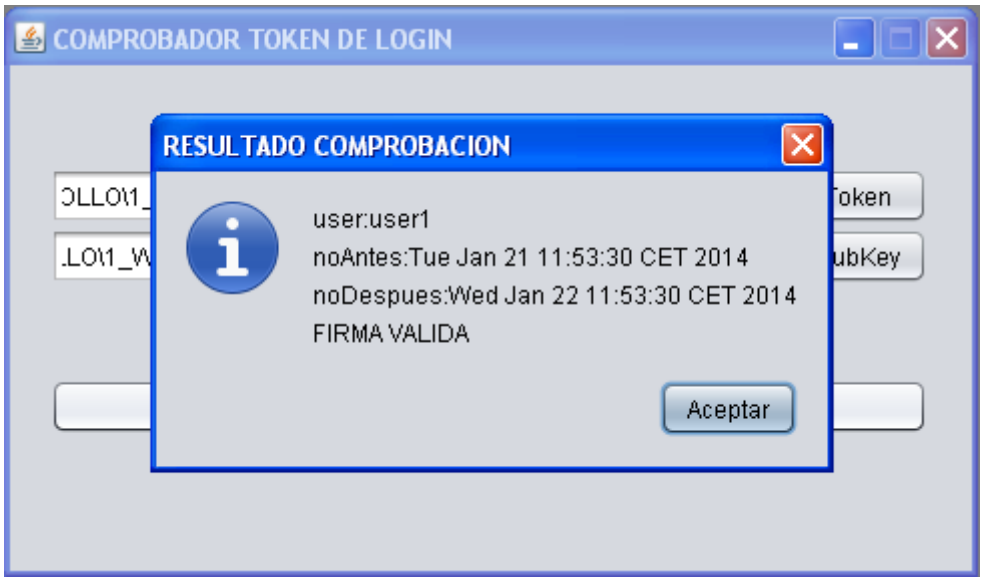


Y la clave pública con la que comprobar la firma.



Pulsamos sobre comprobar

Y si todo es correcto, deberíamos obtener un mensaje como el siguiente, en el cual se especifica el usuario, el periodo de validez y si la firma que aporta confianza al sistema es válida.

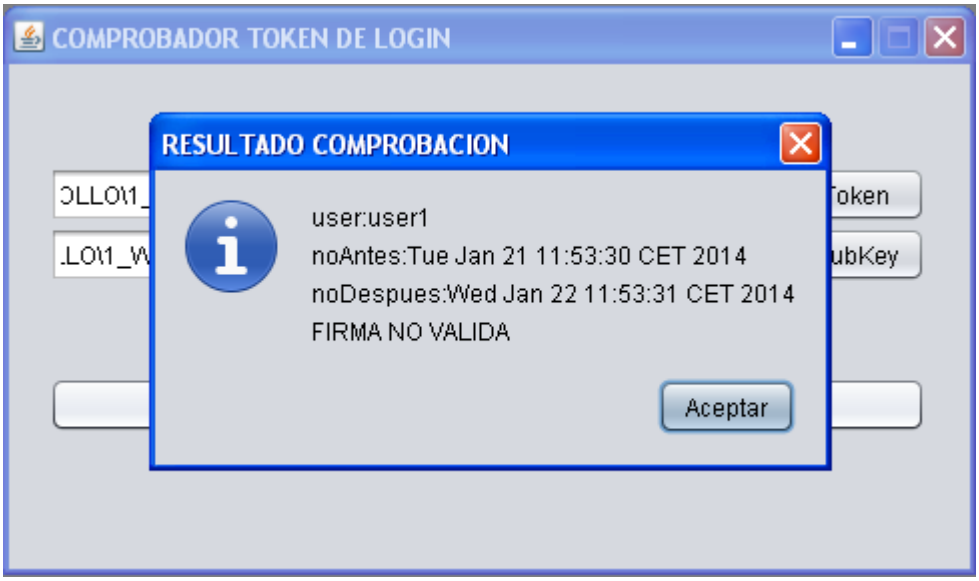


Si realizamos cualquier cambio en el token, como por ejemplo cambiar e periodo de validez en 1 segundo

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <login xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="config/LoginSchema.xsd">
  - <acceso>
      <user>user1</user>
    - <validez>
        <noAntes>1390301610484</noAntes>
        <noDespues>1390388010484</noDespues>
      </validez>

        <firma>4c8ff383bec4e0a95cb7464320a39bc5c3b7c39e023d2c3e4b0899091c371f0e9869e35a75534ccad89a5a131f13b4ec
    </acceso>
  </login>
```

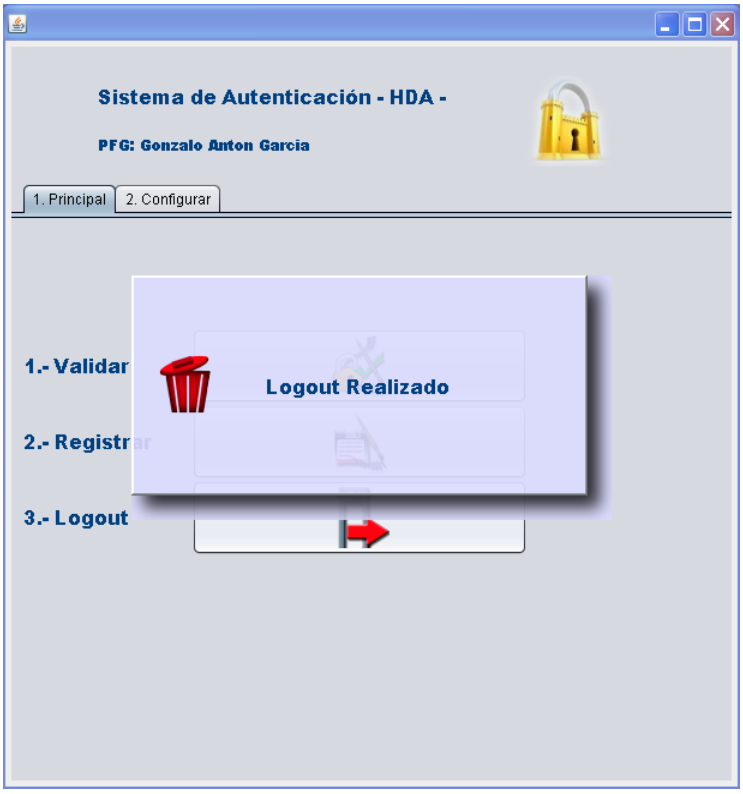
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<login xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="config/LoginSchema.xsd">
- <acceso>
    <user>user1</user>
  - <validez>
      <noAntes>1390301610484</noAntes>
      <noDespues>1390388011484</noDespues>
    </validez>

      <firma>4c8ff383bec4e0a95cb7464320a39bc5c3b7c39e023d2c3e4b0899091c371f0e9869e35a75534ccad89a5a131f13b4e
  </acceso>
</login>
```

La herramienta vemos como delata que la firma de los datos no es correcta y muestra el mensaje FIRMA NO VALIDA.



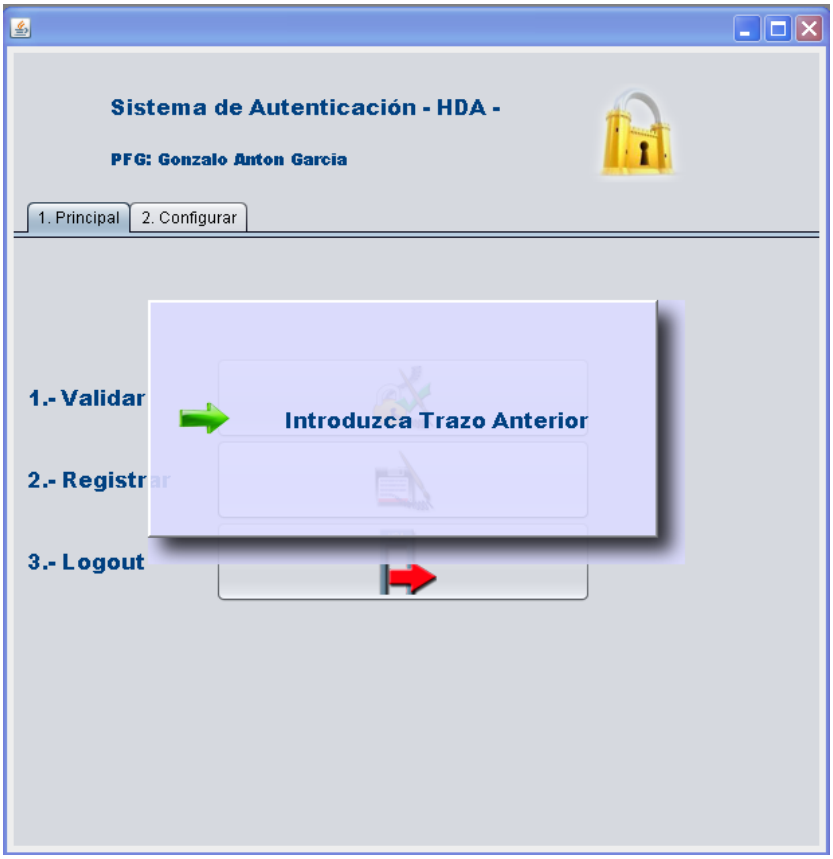
El proceso de Logout se realiza pulsando sobre el boton Logout de la pestaña principal.

El sistema eliminará el fichero de token que deberá ser interpretado como que el usuario no está autenticado.

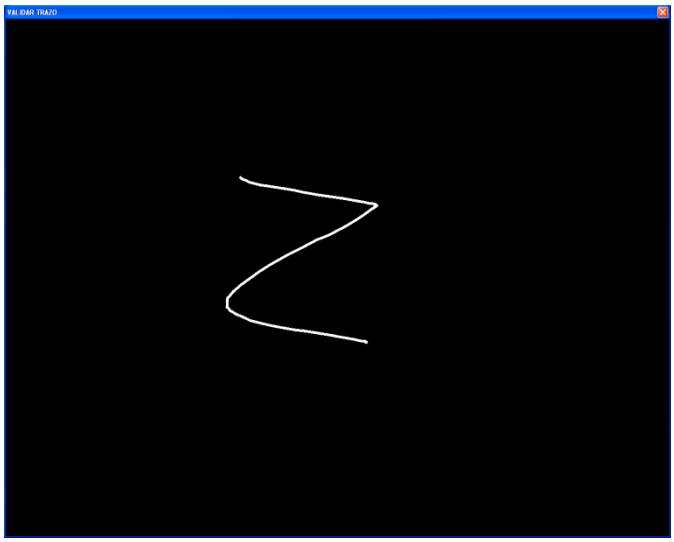


A.2. Cambio de Trazo

Para cambiar el trazo registrado, pulsamos sobre el botón registrar. El sistema no solicitará el trazo previamente almacenado.



Si lo introducimos correctamente



Se nos dará la posibilidad de introducir un nuevo trazo con el que validarnos en el sistema.

El trazo que queremos registrar, debemos repetirlo 3 veces (valor por defecto en configuración) para que el sistema haga cálculos que mejoran sustancialmente la respuesta general del sistema.

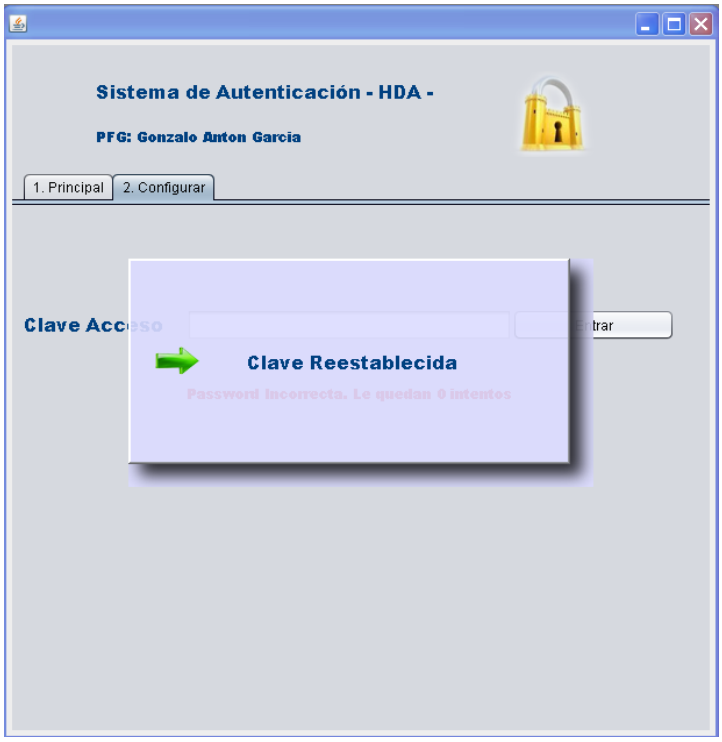
A.3. Bloqueo del sistema

Si introducimos mal la contraseña de acceso al menú del administrador. El sistema un mensaje temporal que bloque el sistema temporalmente (5 segundos). El tiempo de bloquea se incrementa geométricamente con cada intento fallido

Si por algún motivo llegásemos a introducir la contraseña del administrador de forma errónea 5 veces

El sistema, a parte de bloquear durante un tiempo cualquier acción realizada sobre la aplicación

Procedería a realizar un reseteo de la clave del administrador, restableciéndola a sus valores iniciales. De tal manera que se requerirá la ejecución de nuevo por parte del administrador, del software de instalación de PFG_EXTRAS como si se tratase de una instalación nueva.



A parte de esto, el sistema bloqueará de forma permanente el acceso al menú del administrador siendo necesario volver a lanzar la aplicación.

B. Código PFG

B.1. Paquete hda.auth.cifrado

B.1.1. hda.auth.cifrado.ClaveAleatoria.java

```
package hda.auth.cifrado;

public abstract class ClaveAleatoria {
	String algoritmo;
	String fechaGeneracion;
	public abstract String toString();
	public abstract byte[] getClaveCifrar();
	public abstract byte[] getClaveDescifrar();
}
```

B.1.2. hda.auth.cifrado.ClaveAleatoriaAsimetrica.java

```
package hda.auth.cifrado;

import org.bouncycastle.util.encoders.Hex;

public class ClaveAleatoriaAsimetrica extends ClaveAleatoria{
	byte[] clavePublica;
	byte[] clavePrivada;
	byte[] claveCifrar;
	byte[] claveDescifrar;
	public byte[] getClavePublica() {
		return clavePublica;
	}
	public void setClavePublica(byte[] clavePublica) {
		this.clavePublica = clavePublica;
	}
	public byte[] getClavePrivada() {
		return clavePrivada;
	}
	public void setClavePrivada(byte[] clavePrivada) {
		this.clavePrivada = clavePrivada;
	}
	public byte[] getClaveCifrar() {
		return claveCifrar;
	}
	public void setClaveCifrar(byte[] claveCifrar) {
		this.claveCifrar = claveCifrar;
	}
	public byte[] getClaveDescifrar() {
		return claveDescifrar;
	}
	public void setClaveDescifrar(byte[] claveDescifrar) {
		this.claveDescifrar = claveDescifrar;
	}
	public String toString() {
		String cadena = "--BEGIN PRIVATE--\r\n";
		cadena = cadena+new String(Hex.encode(getClavePrivada()));
		cadena = cadena+"\r\n--END PRIVATE--\r\n";
		cadena = cadena+"\r\n--BEGIN PUBLIC--\r\n";
		cadena = cadena+new String(Hex.encode(getClavePublica()));
		cadena = cadena+"\r\n--END PUBLIC--\r\n";
		return cadena;
	}
}
```

B.1.3. hda.auth.cifrado.GeneradorClave.java

```
package hda.auth.cifrado;

import java.util.ArrayList;


public class GeneradorClave{
	public static String generaClaveSecretaUID(){

		ArrayList<String> lista = new ArrayList<String>();

		try{
			lista.add(SystemUniqueId.getBiosSerialNumber());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		try{
			lista.add(SystemUniqueId.getCpuProcessorId());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		try{
			lista.add(SystemUniqueId.getDiskDriveSerialNumber());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		//aquí se podría complicar el algoritmo si quisieramos

		String entropia = "";
		String passwordHashStr = "";
		for(String cadena:lista){
			entropia = SecurityToolBox.getSHA256(cadena+entropia);
		}
		passwordHashStr = entropia;
		return passwordHashStr;

	}

}
```

B.1.4. hda.auth.cifrado.SecurityToolBox

```
package hda.auth.cifrado;

import hda.auth.dao.GenerarClaveException;

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.math.BigInteger;
import java.security.SecureRandom;

import org.bouncycastle.asn1.pkcs.PrivateKeyInfo;
import org.bouncycastle.asn1.x509.SubjectPublicKeyInfo;
import org.bouncycastle.crypto.AsymmetricCipherKeyPair;
import org.bouncycastle.crypto.BufferedAsymmetricBlockCipher;
import org.bouncycastle.crypto.DataLengthException;
import org.bouncycastle.crypto.digests.SHA256Digest;
import org.bouncycastle.crypto.encodings.PKCS1Encoding;
import org.bouncycastle.crypto.engines.AESEngine;
import org.bouncycastle.crypto.engines.RSAEngine;
import org.bouncycastle.crypto.generators.RSAKeyPairGenerator;
import org.bouncycastle.crypto.modes.CBCBlockCipher;
import org.bouncycastle.crypto.paddings.PKCS7Padding;
import org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher;
import org.bouncycastle.crypto.params.AsymmetricKeyParameter;
import org.bouncycastle.crypto.params.KeyParameter;
import org.bouncycastle.crypto.params.RSAKeyGenerationParameters;
import org.bouncycastle.crypto.params.RSAKeyParameters;
import org.bouncycastle.crypto.params.RSAPrivateCrtKeyParameters;
import org.bouncycastle.crypto.util.PrivateKeyFactory;
import org.bouncycastle.crypto.util.PrivateKeyInfoFactory;
import org.bouncycastle.crypto.util.PublicKeyFactory;
import org.bouncycastle.crypto.util.SubjectPublicKeyInfoFactory;
import org.bouncycastle.util.encoders.Hex;

public class SecurityToolBox {

	public static String getSHA256(String texto){
	   SHA256Digest  digest = new SHA256Digest();
	   byte[]  hash = new byte[digest.getDigestSize()];
	   byte[]  bytesTexto = texto.getBytes();
	   digest.update(bytesTexto, 0, bytesTexto.length);
	   digest.doFinal(hash, 0);
	   String hashStr = new String(Hex.encode(hash));
	   return hashStr;
	}

	public static ClaveAleatoria generarParejaClavesAleatoria() throws GenerarClaveException{
		ClaveAleatoriaAsimetrica caa = new ClaveAleatoriaAsimetrica();
		try{

		//Generamos KeyPair
		RSAKeyPairGenerator generator = new RSAKeyPairGenerator();
		generator.init(new RSAKeyGenerationParameters(
			new BigInteger("10001", 16),//exponente debe ser un numero de fermat 65537 es lo habitual
			SecureRandom.getInstance("SHA1PRNG"),//numero aleatorio
			1024,//longitud clave
			80//fiabilidad de numro primo
		  ));

		//Separamos los Objetos del KeyPair
		AsymmetricCipherKeyPair parClaves = generator.generateKeyPair();
		AsymmetricKeyParameter pubKey = parClaves.getPublic();
		AsymmetricKeyParameter privKey = parClaves.getPrivate();
```

```
            //Obtenemos datos para Serializar las claves en formato ASN1
            PrivateKeyInfo privateKeyInfo = PrivateKeyInfoFactory.createPrivateKeyInfo(privKey);
            byte[] clavePrivada = privateKeyInfo.toASN1Primitive().getEncoded();
            SubjectPublicKeyInfo publicKeyInfo =
SubjectPublicKeyInfoFactory.createSubjectPublicKeyInfo(pubKey) ;
            byte[] clavePublica = publicKeyInfo.toASN1Primitive().getEncoded();

            caa.setClavePrivada(clavePrivada);
            caa.setClavePublica(clavePublica);
            } catch (Exception e) {
                  throw new GenerarClaveException("Error generando clave Asimetrica");
            }
      return caa;
      }

      public static void cifrarConClavePrivada(InputStream is, OutputStream os, byte[] key) throws Exception {

            BufferedAsymmetricBlockCipher encryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            try {
                  RSAPrivateCrtKeyParameters privateKey = (RSAPrivateCrtKeyParameters)
PrivateKeyFactory.createKey(key);
                  encryptCipher = new BufferedAsymmetricBlockCipher (new PKCS1Encoding(new RSAEngine()));
                  encryptCipher.init(true, privateKey);

                  //int total = is.available();
                  ByteArrayOutputStream bos = new ByteArrayOutputStream();
                  int noBytesRead = 0;       //number of bytes read from input
                  int len = encryptCipher.getInputBlockSize();
                  buf = new byte[len];
                  while ((noBytesRead = is.read(buf)) >= 0) {
                        try{
                              encryptCipher.processBytes(buf, 0, noBytesRead);
                              obuf = encryptCipher.doFinal();
                              bos.write(obuf, 0, obuf.length);
                              encryptCipher.reset();
                        }catch(DataLengthException dle){
                              obuf = encryptCipher.doFinal();
                              bos.write(obuf, 0, noBytesRead);
                        }
                        //float bosFloat = bos.size();
                        //float totalFloat = total;
                        //float porcentaje = (bosFloat/totalFloat)*100;
                  }
                  os.write(bos.toByteArray());
            }catch(Exception e){
                  e.printStackTrace();
            }

      }
      public static void descifrarConClavePublica(InputStream is, OutputStream os, byte[] key) throws Exception {

            BufferedAsymmetricBlockCipher decryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            try {
                  RSAKeyParameters publicKey = (RSAKeyParameters) PublicKeyFactory.createKey(key);
                  decryptCipher = new BufferedAsymmetricBlockCipher (new PKCS1Encoding(new RSAEngine()));
                  decryptCipher.init(false, publicKey);


                  ByteArrayOutputStream bos = new ByteArrayOutputStream();
                  int noBytesRead = 0;       //number of bytes read from input
                  int len = decryptCipher.getInputBlockSize();
                  buf = new byte[len];
```

```
                while ((noBytesRead = is.read(buf)) >= 0) {
                    try{
                        decryptCipher.processBytes(buf, 0, noBytesRead);
                        obuf = decryptCipher.doFinal();
                        bos.write(obuf, 0, obuf.length);
                        decryptCipher.reset();
                    }catch(DataLengthException dle){
                        obuf = decryptCipher.doFinal();
                        bos.write(obuf, 0, noBytesRead);
                    }
                    //float porcentaje = (bos.size()/total)*100;
                }
                os.write(bos.toByteArray());
            }catch(Exception e){
                e.printStackTrace();
            }

        }

        public static void cifrarConClaveSecreta(InputStream is, OutputStream os, byte[] key) throws Exception{

            PaddedBufferedBlockCipher encryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            //encryptCipher = new PaddedBufferedBlockCipher(new AESEngine());
            encryptCipher = new PaddedBufferedBlockCipher(new CBCBlockCipher(new AESEngine()), new
PKCS7Padding());
            encryptCipher.init(true, new KeyParameter(key));
            try {
                int noBytesRead = 0;        //number of bytes read from input
                int noBytesProcessed = 0;   //number of bytes processed
                while ((noBytesRead = is.read(buf)) >= 0) {
                    noBytesProcessed = encryptCipher.processBytes(buf, 0, noBytesRead, obuf, 0);
                    os.write(obuf, 0, noBytesProcessed);
                }

                noBytesProcessed = encryptCipher.doFinal(obuf, 0);

                os.write(obuf, 0, noBytesProcessed);

                os.flush();
            }
            catch (java.io.IOException e) {
                e.printStackTrace();
            }
        }

        public static void descifrarConClaveSecreta(InputStream is, OutputStream os, byte[] key) throws Exception {

            PaddedBufferedBlockCipher decryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            //decryptCipher =  new PaddedBufferedBlockCipher(new AESEngine());
            decryptCipher =  new PaddedBufferedBlockCipher(new CBCBlockCipher(new AESEngine()),new
PKCS7Padding());
            decryptCipher.init(false, new KeyParameter(key));
            try {
                // Bytes read from in will be decrypted
                // Read in the decrypted bytes from in InputStream and and
                // write them in cleartext to out OutputStream
                int noBytesRead = 0;        //number of bytes read from input
                int noBytesProcessed = 0;   //number of bytes processed
                while ((noBytesRead = is.read(buf)) >= 0) {
                    noBytesProcessed = decryptCipher.processBytes(buf, 0, noBytesRead, obuf, 0);
                    os.write(obuf, 0, noBytesProcessed);
                }
```

```
                noBytesProcessed = decryptCipher.doFinal(obuf, 0);
                os.write(obuf, 0, noBytesProcessed);
                os.flush();
            }
            catch (java.io.IOException e) {
                e.printStackTrace();
            }
        }

}
```

B.1.5. hda.auth.cifrado.SystemUniqueId

```
package hda.auth.cifrado;

import java.util.Scanner;

public class SystemUniqueId {

   public static String getBiosSerialNumber() throws Throwable {
      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "bios", "get", "serialNumber" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;
   }
   public static String getDiskDriveSerialNumber() throws Throwable {

      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "diskdrive", "get", "serialnumber" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;

   }
   public static String getCpuProcessorId() throws Throwable {
      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "cpu", "get", "processorid" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;
      //SerialNumber: CZC7150HB7
      //ProcessorId: BFEBFBFF00000F65
   }
}
```

B.2. Paquete hda.auth.config

B.2.1. hda.auth.config.Config.java

```
package hda.auth.config;


import hda.auth.config.bean.ModuleConfigBean;
import hda.auth.config.parsers.XMLmainConfigParser;

import java.io.File;
import java.util.ArrayList;
import org.apache.logging.log4j.core.config.ConfigurationException;

public class Config {

	private static Config inst;
	private ArrayList<ModuleConfigBean> modules_configurations = new ArrayList<ModuleConfigBean>();

	public static Config getInstance() throws ConfigException {
		if(inst==null){
			inst = new Config();
		}
		return inst;
	}

	public Config() throws ConfigException {
		File configuracion = new File(System.getProperty("mainconfig"));
		if(!configuracion.exists()){
			throw new ConfigurationException("Fichero: "+System.getProperty("mainconfig")+" no
encontrado.");
		}
		//Leemos y procesamos el fichero con la configuracion principal
		XMLmainConfigParser xmlmainconfig = new XMLmainConfigParser(configuracion);
		xmlmainconfig.procesaConfiguracion();

		modules_configurations = xmlmainconfig.getModules_configurations();

	}

	public ArrayList<ModuleConfigBean> getModules_configurations() {
		return modules_configurations;
	}

	public void setModules_configurations(ArrayList<ModuleConfigBean> modules_configurations) {
		this.modules_configurations = modules_configurations;
	}

	public ModuleConfigBean getModuleConfigByName(String module){
		for(ModuleConfigBean moduleAux : modules_configurations){
			if(moduleAux.getName().equals(module)){
				return moduleAux;
			}
		}
		return null;
	}


}
```

B.2.2. hda.auth.config.ConfigException.java

```
package hda.auth.config;

public class ConfigException extends Exception {

	private static final long serialVersionUID = 1L;

	public ConfigException() {
		super();
	}

	public ConfigException(String arg0, Throwable arg1) {
		super(arg0, arg1);
	}

	public ConfigException(String arg0) {
		super(arg0);
	}

	public ConfigException(Throwable arg0) {
		super(arg0);
	}


}
```

B.2.3. hda.auth.config.XMLConfigErrorHandler.java

```
package hda.auth.config;

import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class XMLConfigErrorHandler implements ErrorHandler {

	@Override
	public void error(SAXParseException arg0) throws SAXException {
		System.out.println(arg0.getMessage());
		throw new SAXException(arg0.getMessage());
	}

	@Override
	public void fatalError(SAXParseException arg0) throws SAXException {
		arg0.printStackTrace();
		throw new SAXException(arg0.getMessage());
	}

	@Override
	public void warning(SAXParseException arg0) throws SAXException {
		arg0.printStackTrace();
		throw new SAXException(arg0.getMessage());
	}

}
```

B.3. Paquete hda.auth.config.bean

B.3.1. hda.auth.config.bean.KeyFactoryParameterConfigBean.java

```
package hda.auth.config.bean;

public class KeyFactoryParameterConfigBean {
	private int numerotrazos;
	private String name;
	private String className;
	private boolean enabled;

	public int getNumerotrazos() {
		return numerotrazos;
	}
	public void setNumerotrazos(int numerotrazos) {
		this.numerotrazos = numerotrazos;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getClassName() {
		return className;
	}
	public void setClassName(String className) {
		this.className = className;
	}
	public boolean isEnabled() {
		return enabled;
	}
	public void setEnabled(boolean enabled) {
		this.enabled = enabled;
	}

}
```

B.3.2. hda.auth.config.bean.LockFactoryParameterConfigBean.java

```
package hda.auth.config.bean;

public class LockFactoryParameterConfigBean extends KeyFactoryParameterConfigBean {
	private int tolerance;

	public int getTolerance() {
		return tolerance;
	}

	public void setTolerance(int tolerance) {
		this.tolerance = tolerance;
	}
}
```

B.3.3. hda.auth.config.bean.ModuleConfigBean.java

```
package hda.auth.config.bean;

public class ModuleConfigBean {
	private String name;
	private String version;
	private String configFile;

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getVersion() {
		return version;
	}
	public void setVersion(String version) {
		this.version = version;
	}
	public String getConfigFile() {
		return configFile;
	}
	public void setConfigFile(String configFile) {
		this.configFile = configFile;
	}
}
```

B.4. Paquete hda.auth.config.parsers

B.4.1. hda.auth.config.parsers.LoginGenerationException.java

```
package hda.auth.config.parsers;

public class LoginGenerationException extends Exception {

	private static final long serialVersionUID = 1L;

}
```

B.4.2. hda.auth.config.parsers.XMLkeyFactoryParser.java

```
package hda.auth.config.parsers;

import hda.auth.config.XMLConfigErrorHandler;
import hda.auth.config.bean.KeyFactoryParameterConfigBean;
import hda.auth.keys.KeyFactoryConfigException;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class XMLkeyFactoryParser extends DefaultHandler {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private static final String JAXP_SCHEMA_LANGUAGE =
"http://java.sun.com/xml/jaxp/properties/schemaLanguage";//
	private static final String W3C_XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";//

	private File file;
	private SAXParser saxparser;
	private KeyFactoryParameterConfigBean key_param_config_aux;
	private int numerotrazos = 1;
	private ArrayList<KeyFactoryParameterConfigBean> key_params_configurations = new
ArrayList<KeyFactoryParameterConfigBean>();
	public StringBuffer textBuffer = new StringBuffer("INICIALIZADO");

	public XMLkeyFactoryParser(File file) {
		this.file=file;
	}

	@Override
	public final void startElement(String ns,String name,String qname,Attributes attrs) throws SAXException{

		if (qname.equals("numerotrazos")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("parameter")) {
			textBuffer.delete(0,textBuffer.length());
			key_param_config_aux = new KeyFactoryParameterConfigBean();
		}

		if (qname.equals("name")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("class")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("status")) {
```

```
                textBuffer.delete(0,textBuffer.length());
            }

        }

        @Override
        public final void endElement(String ns, String name, String qname){
            if (qname.equals("numerotrazos")) {
                numerotrazos = Integer.parseInt(textBuffer.toString());
            }
            if(qname.equals("name")){
                key_param_config_aux.setName(textBuffer.toString());
            }
            if(qname.equals("class")){
                key_param_config_aux.setClassName(textBuffer.toString());
            }
            if(qname.equals("status")){
                key_param_config_aux.setEnabled( (textBuffer.toString().equals("enabled")) ? true : false );
            }
            if(qname.equals("parameter")){
                key_params_configurations.add(key_param_config_aux);
            }
        }

        public void procesaConfiguracion() throws KeyFactoryConfigException{

            try {
                //Generamos SAXParser
                SAXParserFactory sax_parser_factory = SAXParserFactory.newInstance();


                //validamos el XML
                sax_parser_factory.setValidating(true);
                sax_parser_factory.setNamespaceAware(true);
                saxparser = sax_parser_factory.newSAXParser();
                saxparser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);

                //validamos
                XMLReader reader = saxparser.getXMLReader();
                reader.setErrorHandler(new XMLConfigErrorHandler());

                logger.debug(file.getAbsolutePath());
                reader.parse(new InputSource(file.getPath()));

                //parseamos el XML
                saxparser.parse(file.getPath(), this);

            } catch (SAXException e) {
                throw new KeyFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (IOException e) {
                throw new KeyFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (ParserConfigurationException e) {
                throw new KeyFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            }
        }

        @Override
         public void characters(char[] arg0, int arg1, int arg2) throws org.xml.sax.SAXException{

              String s = new String(arg0, arg1, arg2);
              s = s.replace("&", "&amp;");
              if (textBuffer == null){
                   textBuffer = new StringBuffer(s);
              }else{
                   textBuffer.append(s);
              }
         }
```

```
	public ArrayList<KeyFactoryParameterConfigBean> getKey_params_configurations() {
		return key_params_configurations;
	}

	public void setKey_params_configurations(ArrayList<KeyFactoryParameterConfigBean>
key_params_configurations) {
		this.key_params_configurations = key_params_configurations;
	}

	public int getNumerotrazos() {
		return numerotrazos;
	}

	public void setNumerotrazos(int numerotrazos) {
		this.numerotrazos = numerotrazos;
	}
}
```

B.4.3. hda.auth.config.parsers.XMLlockFactoryParser.java

```
package hda.auth.config.parsers;


import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.config.XMLConfigErrorHandler;
import hda.auth.config.bean.LockFactoryParameterConfigBean;
import hda.auth.dao.ExportarConfigException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class XMLlockFactoryParser extends DefaultHandler {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private static final String JAXP_SCHEMA_LANGUAGE =
"http://java.sun.com/xml/jaxp/properties/schemaLanguage";//
	private static final String W3C_XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";//

	private File file;
	private SAXParser saxparser;
	private LockFactoryParameterConfigBean lock_param_config_aux;
	private int numerotrazos = 1;
	private String dispositivo = "";
	private int horasvalidez = 24;
	private ArrayList<LockFactoryParameterConfigBean> lock_params_configurations = new
ArrayList<LockFactoryParameterConfigBean>();
	public StringBuffer textBuffer = new StringBuffer("INICIALIZADO");

	public XMLlockFactoryParser(File file) {
		this.file=file;
	}

	@Override
	public final void startElement(String ns,String name,String qname,Attributes attrs) throws SAXException{

		if (qname.equals("numerotrazos")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("dispositivo")) {
			textBuffer.delete(0,textBuffer.length());
```

```
        }

        if (qname.equals("horasvalidez")) {
            textBuffer.delete(0,textBuffer.length());
        }

        if (qname.equals("parameter")) {
            textBuffer.delete(0,textBuffer.length());
            lock_param_config_aux = new LockFactoryParameterConfigBean();
        }

        if (qname.equals("name")) {
            textBuffer.delete(0,textBuffer.length());
        }

        if (qname.equals("class")) {
            textBuffer.delete(0,textBuffer.length());
        }
        if (qname.equals("tolerance")) {
            textBuffer.delete(0,textBuffer.length());
        }
        if (qname.equals("status")) {
            textBuffer.delete(0,textBuffer.length());
        }

    }

    @Override
    public final void endElement(String ns, String name, String qname){
        if (qname.equals("numerotrazos")) {
            numerotrazos = Integer.parseInt(textBuffer.toString());
        }
        if (qname.equals("dispositivo")) {
            dispositivo = textBuffer.toString();
        }
        if (qname.equals("horasvalidez")) {
            horasvalidez = Integer.parseInt(textBuffer.toString());
        }
        if(qname.equals("name")){
            lock_param_config_aux.setName(textBuffer.toString());
        }
        if(qname.equals("class")){
            lock_param_config_aux.setClassName(textBuffer.toString());
        }
        if(qname.equals("tolerance")){
            lock_param_config_aux.setTolerance(Integer.parseInt(textBuffer.toString()));
        }
        if(qname.equals("status")){
            lock_param_config_aux.setEnabled( (textBuffer.toString().equals("enabled")) ? true : false );
        }
        if(qname.equals("parameter")){
            lock_params_configurations.add(lock_param_config_aux);
        }
    }

    public void procesaConfiguracion() throws LockFactoryConfigException{

        try {
            //Generamos SAXParser
            SAXParserFactory sax_parser_factory = SAXParserFactory.newInstance();


            //validamos el XML
            sax_parser_factory.setValidating(true);
            sax_parser_factory.setNamespaceAware(true);
            saxparser = sax_parser_factory.newSAXParser();
            saxparser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
```

```
                //validamos
                XMLReader reader = saxparser.getXMLReader();
                reader.setErrorHandler(new XMLConfigErrorHandler());

                logger.debug(file.getAbsolutePath());
                reader.parse(new InputSource(file.getPath()));

                //parseamos el XML
                saxparser.parse(file.getPath(), this);

            } catch (SAXException e) {
                throw new LockFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (IOException e) {
                throw new LockFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (ParserConfigurationException e) {
                throw new LockFactoryConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            }
        }

        public void guardaConfiguracion() throws LockFactoryConfigException{
            try{
                DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

                //ROOT
                Document doc = docBuilder.newDocument();
                Element configElement = doc.createElement("config");
                configElement.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema-instance");
                configElement.setAttribute("xsi:noNamespaceSchemaLocation",
Config.getInstance().getModuleConfigByName("DirSchemas").getConfigFile()+"/LockFactoryConfigSchema.xsd");
                doc.appendChild(configElement);

                //INI-numeroTrazos
                Element numerotrazos = doc.createElement("numerotrazos");

        numerotrazos.appendChild(doc.createTextNode(Integer.toString(LockFactory.getInstance().getNumero_trazos()
)));
                configElement.appendChild(numerotrazos);
                //FIN-numeroTrazos

                //INI-dispositivo
                Element dispositivo = doc.createElement("dispositivo");
                dispositivo.appendChild(doc.createTextNode(LockFactory.getInstance().getDispositivo()));
                configElement.appendChild(dispositivo);
                //FIN-dispositivo

                //INI-horasValidez
                Element horasvalidez = doc.createElement("horasvalidez");

        horasvalidez.appendChild(doc.createTextNode(Integer.toString(LockFactory.getInstance().getHoras_validez()))
);
                configElement.appendChild(horasvalidez);
                //FIN-dispositivo

                for(LockFactoryParameterConfigBean
lock_param_config_aux:LockFactory.getInstance().getLock_params_configurations()){
                    //INI-parametro
                    Element parametro = doc.createElement("parameter");
                    //INI-name
                    Element name = doc.createElement("name");
                    name.appendChild(doc.createTextNode(lock_param_config_aux.getName()));
                    parametro.appendChild(name);
                    //FIN-name

                    //INI-class
                    Element clase = doc.createElement("class");
```

```
                    clase.appendChild(doc.createTextNode(lock_param_config_aux.getClassName()));
                    parametro.appendChild(clase);
                    //FIN-class

                    //INI-tolerance
                    Element tolerance = doc.createElement("tolerance");

    tolerance.appendChild(doc.createTextNode(Integer.toString(lock_param_config_aux.getTolerance())));
                    parametro.appendChild(tolerance);
                    //FIN-tolerance

                    //INI-status
                    Element status = doc.createElement("status");
                    if(lock_param_config_aux.isEnabled()){
                        status.appendChild(doc.createTextNode("enabled"));
                    }else{
                        status.appendChild(doc.createTextNode("disabled"));
                    }
                    //FIN-status
                    parametro.appendChild(status);
                    configElement.appendChild(parametro);
                }


                ManejadorFicheros.exportar(doc, new
File(Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile()));


            } catch (ParserConfigurationException pce) {
                pce.printStackTrace();
            } catch (ExportarConfigException e) {
                e.printStackTrace();
            } catch (ConfigException e) {
                e.printStackTrace();
            }
        }

        @Override
         public void characters(char[] arg0, int arg1, int arg2) throws org.xml.sax.SAXException{

             String s = new String(arg0, arg1, arg2);
             s = s.replace("&", "&amp;");
             if (textBuffer == null){
                 textBuffer = new StringBuffer(s);
             }else{
                 textBuffer.append(s);
             }
         }

        public ArrayList<LockFactoryParameterConfigBean> getLock_params_configurations() {
            return lock_params_configurations;
        }

        public void setLock_params_configurations(ArrayList<LockFactoryParameterConfigBean>
lock_params_configurations) {
            this.lock_params_configurations = lock_params_configurations;
        }

        public int getNumerotrazos() {
            return numerotrazos;
        }

        public void setNumerotrazos(int numerotrazos) {
            this.numerotrazos = numerotrazos;
        }

        public String getDispositivo() {
```

```
        return dispositivo;
    }

    public void setDispositivo(String dispositivo) {
        this.dispositivo = dispositivo;
    }

    public int getHorasvalidez() {
        return horasvalidez;
    }

    public void setHorasvalidez(int horasvalidez) {
        this.horasvalidez = horasvalidez;
    }
}
```

B.4.4. hda.auth.config.parsers.XMLloginParser.java

```
package hda.auth.config.parsers;

import hda.auth.cifrado.SecurityToolBox;
import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.dao.ExportarConfigException;
import hda.auth.dao.ImportarClaveException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.locks.LockFactory;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.Calendar;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.bouncycastle.util.encoders.Hex;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;

public class XMLloginParser {


	public void generaLogin() throws LoginGenerationException{
		try{
			DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
			DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

			//ROOT
			Document doc = docBuilder.newDocument();
			Element login = doc.createElement("login");
			login.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema-instance");
			login.setAttribute("xsi:noNamespaceSchemaLocation", "config/LoginSchema.xsd");
			doc.appendChild(login);

			long duracion = (1000*3600*LockFactory.getInstance().getHoras_validez());
			long current_timestamp = Calendar.getInstance().getTimeInMillis();
			long future_timestamp = current_timestamp + duracion;

			String usuarioStr = "user1";
			String noAntesStr = ""+current_timestamp;
			String noDespuesStr = ""+future_timestamp;

			//Firmamos la información de Login
			String textoClaro = usuarioStr+noAntesStr+noDespuesStr;
			String hashStr = SecurityToolBox.getSHA256(textoClaro);
			ByteArrayInputStream bis = new ByteArrayInputStream(hashStr.getBytes());
			ByteArrayOutputStream bos = new ByteArrayOutputStream();
			SecurityToolBox.cifrarConClavePrivada(bis, bos, ManejadorFicheros.importarClavePrivada());
			String firmaStr = new String(Hex.encode(bos.toByteArray()));

			//INI-acceso
			Element acceso = doc.createElement("acceso");

			//INI-user
			Element user = doc.createElement("user");
			user.appendChild(doc.createTextNode(usuarioStr));
			acceso.appendChild(user);
```

```
                //FIN-user

                //INI-validez
                Element validez = doc.createElement("validez");

                //INI-noAntes
                Element noAntes = doc.createElement("noAntes");
                noAntes.appendChild(doc.createTextNode(noAntesStr));
                validez.appendChild(noAntes);
                //FIN-noAntes

                //INI-noDespues
                Element noDespues = doc.createElement("noDespues");
                noDespues.appendChild(doc.createTextNode(noDespuesStr));
                validez.appendChild(noDespues);
                //FIN-noAntes

                acceso.appendChild(validez);
                //FIN-validez

                //INI-firma
                Element firma = doc.createElement("firma");
                firma.appendChild(doc.createTextNode(firmaStr));
                acceso.appendChild(firma);
                //FIN-firma

                login.appendChild(acceso);
                //FIN-acceso

                ManejadorFicheros.exportar(doc, new
File(Config.getInstance().getModuleConfigByName("LoginObject").getConfigFile()));

            } catch (ParserConfigurationException pce) {
                pce.printStackTrace();
            } catch (ExportarConfigException e) {
                e.printStackTrace();
            } catch (ConfigException e) {
                e.printStackTrace();
            } catch (ImportarClaveException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        public void addLogin() throws LoginGenerationException{
            try{
                File fXmlFile = new
File(Config.getInstance().getModuleConfigByName("LoginObject").getConfigFile());
                DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                Document doc = dBuilder.parse(fXmlFile);

                //INI-acceso
                Element acceso = doc.createElement("acceso");

                //INI-user
                Element user = doc.createElement("user");
                user.appendChild(doc.createTextNode(""));
                acceso.appendChild(user);
                //FIN-user

                //INI-validez
                Element validez = doc.createElement("validez");

                //INI-noAntes
                Element noAntes = doc.createElement("noAntes");
```

```
                noAntes.appendChild(doc.createTextNode(""));
                validez.appendChild(noAntes);
                //FIN-noAntes

                //INI-noDespues
                Element noDespues = doc.createElement("noDespues");
                noDespues.appendChild(doc.createTextNode(""));
                validez.appendChild(noDespues);
                //FIN-noAntes

                acceso.appendChild(validez);
                //FIN-validez

                //INI-firma
                Element firma = doc.createElement("user");
                firma.appendChild(doc.createTextNode(""));
                acceso.appendChild(firma);
                //FIN-firma

                doc.getDocumentElement().appendChild(acceso);
                //FIN-acceso

                ManejadorFicheros.exportar(doc, new
File(Config.getInstance().getModuleConfigByName("LoginObject").getConfigFile()));

            } catch (ParserConfigurationException pce) {
                pce.printStackTrace();
            } catch (ExportarConfigException e) {
                e.printStackTrace();
            } catch (ConfigException e) {
                e.printStackTrace();
            } catch (SAXException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

B.4.5. hda.auth.config.parsers.XMLmainConfigParser.java

```
package hda.auth.config.parsers;

import hda.auth.config.ConfigException;
import hda.auth.config.XMLConfigErrorHandler;
import hda.auth.config.bean.ModuleConfigBean;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class XMLmainConfigParser extends DefaultHandler {

	private static final String JAXP_SCHEMA_LANGUAGE =
"http://java.sun.com/xml/jaxp/properties/schemaLanguage";//
	private static final String W3C_XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";//

	private File file;
	private SAXParser saxparser;
	private ModuleConfigBean module_config_aux;
	private ArrayList<ModuleConfigBean> modules_configurations = new ArrayList<ModuleConfigBean>();
	public StringBuffer textBuffer = new StringBuffer("INICIALIZADO");

	public XMLmainConfigParser(File file) {
		this.file=file;
	}

	@Override
	public final void startElement(String ns,String name,String qname,Attributes attrs) throws SAXException{

		if (qname.equals("module")) {
			textBuffer.delete(0,textBuffer.length());
			module_config_aux = new ModuleConfigBean();
		}

		if (qname.equals("name")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("version")) {
			textBuffer.delete(0,textBuffer.length());
		}

		if (qname.equals("configfile")) {
			textBuffer.delete(0,textBuffer.length());
		}

	}

	@Override
	public final void endElement(String ns, String name, String qname){

		if(qname.equals("name")){
			module_config_aux.setName(textBuffer.toString());
```

```
            }
            if(qname.equals("version")){
                module_config_aux.setVersion(textBuffer.toString());
            }
            if(qname.equals("configfile")){
                module_config_aux.setConfigFile(textBuffer.toString());
            }
            if(qname.equals("module")){
                modules_configurations.add(module_config_aux);
            }
        }

        public void procesaConfiguracion() throws ConfigException{

            try {
                //Generamos SAXParser
                SAXParserFactory sax_parser_factory = SAXParserFactory.newInstance();


                //validamos el XML
                sax_parser_factory.setValidating(true);
                sax_parser_factory.setNamespaceAware(true);
                saxparser = sax_parser_factory.newSAXParser();
                saxparser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);

                //validamos
                XMLReader reader = saxparser.getXMLReader();
                reader.setErrorHandler(new XMLConfigErrorHandler());

                reader.parse(new InputSource(file.getPath()));

                //parseamos el XML
                saxparser.parse(file.getPath(), this);

            } catch (SAXException e) {
                throw new ConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (IOException e) {
                throw new ConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            } catch (ParserConfigurationException e) {
                throw new ConfigException("Error Configuracion: <"+file+"> causa: "+e.getMessage());
            }
        }

        @Override
         public void characters(char[] arg0, int arg1, int arg2) throws org.xml.sax.SAXException{

             String s = new String(arg0, arg1, arg2);
             s = s.replace("&", "&amp;");
             if (textBuffer == null){
                 textBuffer = new StringBuffer(s);
             }else{
                 textBuffer.append(s);
             }
         }

        public ArrayList<ModuleConfigBean> getModules_configurations() {
            return modules_configurations;
        }

        public void setModules_configurations(ArrayList<ModuleConfigBean> modules_configurations) {
            this.modules_configurations = modules_configurations;
        }

}
```

B.5. Paquete hda.auth.dao

B.5.1. hda.auth.dao.CerrarFlujoException.java

```
package hda.auth.dao;

public class CerrarFlujoException extends Exception {

	private static final long serialVersionUID = 1L;

	public CerrarFlujoException(String arg0) {
		super(arg0);
	}
}
```

B.5.2. hda.auth.dao.ExportarClaveException.java

```
package hda.auth.dao;

public class ExportarClaveException extends Exception{

	private static final long serialVersionUID = 1L;

	public ExportarClaveException(String arg0) {
		super(arg0);
	}
}
```

B.5.3. hda.auth.dao.ExportarConfigException.java

```
package hda.auth.dao;

public class ExportarConfigException extends Exception{

	private static final long serialVersionUID = 1L;

	public ExportarConfigException(String arg0) {
		super(arg0);
	}
}
```

B.5.4. hda.auth.dao.ExportarException.java

```
package hda.auth.dao;

public class ExportarException extends Exception{

	private static final long serialVersionUID = 1L;

	public ExportarException(String arg0) {
		super(arg0);
	}
}
```

B.5.5. hda.auth.dao.ExportarLockException.java

```
package hda.auth.dao;

public class ExportarLockException extends Exception{

	private static final long serialVersionUID = 1L;

	public ExportarLockException(String arg0) {
		super(arg0);
	}
}
```

B.5.6. hda.auth.dao.ExportarLoginException.java

```
package hda.auth.dao;

public class ExportarLoginException extends Exception{

	private static final long serialVersionUID = 1L;

	public ExportarLoginException(String arg0) {
		super(arg0);
	}
}
```

B.5.7. hda.auth.dao.GenerarClaveException.java

```
package hda.auth.dao;

public class GenerarClaveException extends Exception {

	private static final long serialVersionUID = 1L;

	public GenerarClaveException(String arg0) {
		super(arg0);
	}
}
```

B.5.8. hda.auth.dao.ImportarClaveException.java

```
package hda.auth.dao;

public class ImportarClaveException extends Exception {

	private static final long serialVersionUID = 1L;

	public ImportarClaveException(String arg0) {
		super(arg0);
	}
}
```

B.5.9. hda.auth.dao.ManejadorFicheros.java

```
package hda.auth.dao;

import hda.auth.cifrado.ClaveAleatoria;
import hda.auth.cifrado.ClaveAleatoriaAsimetrica;
import hda.auth.cifrado.GeneradorClave;
import hda.auth.cifrado.SecurityToolBox;
import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.locks.Lock;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectOutputStream;

import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.bouncycastle.util.encoders.Hex;
import org.w3c.dom.Document;

/**
 *
 * @author Gonzalo Antón García
 *
 */
public class ManejadorFicheros extends ManejadorRecursos{

	public static byte[] importarClavePrivada() throws ImportarClaveException{
		ByteArrayOutputStream baos = new ByteArrayOutputStream();
		FileInputStream fis = null;
		BufferedInputStream bais = null;
		try {
			int bytesLeidos;
			byte[] buffer = new byte[1024];
			fis = new
FileInputStream(Config.getInstance().getModuleConfigByName("PrivateKey").getConfigFile());
			bais = new BufferedInputStream(fis);

			while( (bytesLeidos = bais.read(buffer)) > 0 ){
				baos.write(buffer,0,bytesLeidos);
				baos.flush();
			}
			//INI-Descifrar clave privada
			ByteArrayInputStream bis = new ByteArrayInputStream(Hex.decode((baos.toByteArray())));
			ByteArrayOutputStream bos = new ByteArrayOutputStream();
			SecurityToolBox.descifrarConClaveSecreta(bis, bos,
Hex.decode(GeneradorClave.generaClaveSecretaUID()));
			//FIN-Descifrar clave privada
			return bos.toByteArray();

		} catch (IOException e) {
			throw new ImportarClaveException("Error al importar el fichero de clave privada: "+e.getMessage());
		} catch (ConfigException e) {
			throw new ImportarClaveException("Error al importar el fichero de clave privada: "+e.getMessage());
```

```
            }catch (Exception e) {
                throw new ImportarClaveException("Error al importar el fichero de clave privada: "+e.getMessage());
            } finally{
                try {
                    if(bais!=null){
                        bais.close();
                    }
                } catch (IOException e) {
                    throw new ImportarClaveException("Error al cerrar fichero de clave privada: "+e.getMessage());
                }
            }
        }

        public static byte[] importarClaveAdmin() throws ImportarClaveException{
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            FileInputStream fis = null;
            BufferedInputStream bais = null;
            try {
                int bytesLeidos;
                byte[] buffer = new byte[1024];
                fis = new
FileInputStream(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile());
                bais = new BufferedInputStream(fis);

                while( (bytesLeidos = bais.read(buffer)) > 0 ){
                    baos.write(buffer,0,bytesLeidos);
                }
                //INI-Descifrar clave privada
                ByteArrayInputStream bis = new ByteArrayInputStream(Hex.decode(baos.toByteArray()));
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                SecurityToolBox.descifrarConClaveSecreta(bis, bos,
Hex.decode(GeneradorClave.generaClaveSecretaUID()));
                //FIN-Descifrar clave privada
                return bos.toByteArray();

            } catch (IOException e) {
                throw new ImportarClaveException("Error al importar el fichero de clave secreta Admin:
"+e.getMessage());
            } catch (ConfigException e) {
                throw new ImportarClaveException("Error al importar el fichero de clave secreta Admin:
"+e.getMessage());
            }catch (Exception e) {
                throw new ImportarClaveException("Error al importar el fichero de clave secreta Admin:
"+e.getMessage());
            } finally{
                try {
                    if(bais!=null){
                        bais.close();
                    }
                } catch (IOException e) {
                    throw new ImportarClaveException("Error al cerrar fichero de clave privada: "+e.getMessage());
                }
            }
        }

        public static void exportar(ClaveAleatoria clave,File file) throws ExportarClaveException{
            if(clave instanceof ClaveAleatoriaAsimetrica){
                String filePathPrivate = "";
                String filePathPublic = "";
                if(file.getName().contains(".key")){
                    try {
                        filePathPrivate =
Config.getInstance().getModuleConfigByName("PrivateKey").getConfigFile();
                    } catch (ConfigException e) {
                        filePathPrivate = "data"+File.separator+"private.key";
                    }
                    filePathPublic = file.getPath().substring(0,file.getPath().indexOf(file.getName()))+file.getName();
```

```
                }else{
                    try {
                        filePathPrivate =
Config.getInstance().getModuleConfigByName("PrivateKey").getConfigFile();
                    } catch (ConfigException e) {
                        filePathPrivate = "data"+File.separator+"private.key";
                    }
                    filePathPublic =
file.getPath().substring(0,file.getPath().indexOf(file.getName()))+file.getName()+".key";
                }
                FileWriter  fosPriv = null;
                FileWriter  fosPub = null;
                try {
                    fosPriv = new FileWriter(filePathPrivate);
                    fosPub = new FileWriter(filePathPublic);
                    //INI-Cifrar clave privada
                    ByteArrayInputStream bis = new
ByteArrayInputStream(((ClaveAleatoriaAsimetrica)clave).getClavePrivada());
                    ByteArrayOutputStream bos = new ByteArrayOutputStream();
                    SecurityToolBox.cifrarConClaveSecreta(bis, bos,
Hex.decode(GeneradorClave.generaClaveSecretaUID()));
                    //FIN-Cifrar clave privada
                    fosPriv.write(new String(Hex.encode(bos.toByteArray())));
                    fosPub.write(new String(Hex.encode(((ClaveAleatoriaAsimetrica)clave).getClavePublica())));
                    fosPriv.flush();
                    fosPub.flush();
                } catch (IOException e) {
                    throw new ExportarClaveException("Error al exportar el fichero:"+file.getName());
                } catch (Exception e) {
                    throw new ExportarClaveException("Error al exportar el fichero:"+file.getName());
                } finally{
                    try {
                        if(fosPriv!=null){
                            fosPriv.close();
                        }
                        if(fosPub!=null){
                            fosPub.close();
                        }
                    } catch (IOException e) {
                        throw new ExportarClaveException("Error al cerrar el fichero:"+file.getName());
                    }
                }
            }
        }

        public static void exportar(Document doc, File file) throws ExportarConfigException {
            try{
                // write the content into xml file
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource source = new DOMSource(doc);
                StreamResult result = new StreamResult(file);
                transformer.transform(source, result);
            } catch (TransformerException tfe) {
                throw new ExportarConfigException(tfe.getMessage());
            }
        }

        public static void exportar(Lock lock, File file) throws ExportarLockException {
            try{
            FileOutputStream fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(lock);
            oos.close();
            }catch(IOException ioe){
                throw new ExportarLockException(ioe.getMessage());
            }
```

```
	}

	public static void exportar(String cadena, File file) throws ExportarException{
		try{
			FileWriter fr = new FileWriter(file);
			try {
				//INI-Cifrar clave privada
				ByteArrayInputStream bis = new ByteArrayInputStream(cadena.getBytes());
				ByteArrayOutputStream bos = new ByteArrayOutputStream();
				SecurityToolBox.cifrarConClaveSecreta(bis, bos,
Hex.decode(GeneradorClave.generaClaveSecretaUID()));
				//FIN-Cifrar clave privada
				fr.write(new String(Hex.encode(bos.toByteArray())));
				fr.flush();
			} catch (IOException e) {
				throw new ExportarException("Error al exportar el fichero:"+file.getName());
			} catch (Exception e) {
				throw new ExportarException("Error al exportar el fichero:"+file.getName());
			} finally{
				try {
					if(fr!=null){
						fr.close();
					}
				} catch (IOException e) {
					throw new ExportarException("Error al cerrar el fichero:"+file.getName());
				}
			}
		}catch(IOException ioe){
			throw new ExportarException(ioe.getMessage());
		}
	}


}
```

B.5.10. hda.auth.dao.ManejadorRecursos.java

```
package hda.auth.dao;

public class ManejadorRecursos {

}
```

B.5.11. hda.auth.dao.ObtenerFlujoException.java

```
package hda.auth.dao;

public class ObtenerFlujoException extends Exception {

	private static final long serialVersionUID = 1L;

	public ObtenerFlujoException(String arg0) {
		super(arg0);
	}
}
```

B.6. Paquete hda.auth.gui

B.6.1. hda.auth.gui.DialogAdminPasswordSetup.java

```
package hda.auth.gui;


import hda.auth.gui.listeners.AdminPasswordSetupListener;

import java.io.Serializable;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class DialogAdminPasswordSetup extends JDialog implements Serializable{

	private static final long serialVersionUID = 1L;
	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private JFrame framePadre;
	private JPanel panelFormulario;
	private JLabel claveAccesoLabel;
	private JPasswordField claveAccesoField;
	private JLabel claveAccesoRepetirLabel;
	private JPasswordField claveAccesoRepetirField;
	private JButton cambiarBoton;
	private JButton cancelarBoton;
	private JLabel mensajeInformativoLabel;
```

```
    public DialogAdminPasswordSetup(JFrame framePadre){
        //framePadre.setEnabled(false);
        super(framePadre,true);
        this.framePadre = framePadre;
        initComponents();
    }

    private void initComponents(){
        logger.debug("iniciamos el JDialog AdminPasswordSetup");
        setSize(450,250);
        setLocation((framePadre.getX())+(framePadre.getWidth()/2)-(getWidth()/2) ,
(framePadre.getY())+(framePadre.getHeight()/2)-(getHeight()/2));
        setTitle("NUEVA PASSWORD");
        AdminPasswordSetupListener adminPasswordSetupListener = new AdminPasswordSetupListener(this);
        addWindowListener(adminPasswordSetupListener);

        panelFormulario = new JPanel();
        FormLayout adminPasswordLayout = new FormLayout("max(p;140), max(p;100), max(p;100),
max(p;100)","max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30)");
        panelFormulario.setLayout(adminPasswordLayout);
        panelFormulario.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

        {
            claveAccesoLabel = new JLabel();
            claveAccesoLabel.setText("Nueva clave");
            claveAccesoLabel.setForeground(new java.awt.Color(0,64,128));
            claveAccesoLabel.setFont(new java.awt.Font("Arial Black",0,16));
            panelFormulario.add(claveAccesoLabel, new CellConstraints(1, 2, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            claveAccesoField = new JPasswordField(JPasswordField.WHEN_FOCUSED);
            claveAccesoField.setName("password");
            panelFormulario.add(claveAccesoField, new CellConstraints(2, 2, 2, 1, CellConstraints.FILL,
CellConstraints.CENTER));

        }
        {
            claveAccesoRepetirLabel = new JLabel();
            claveAccesoRepetirLabel.setText("Repita Clave");
            claveAccesoRepetirLabel.setForeground(new java.awt.Color(0,64,128));
            claveAccesoRepetirLabel.setFont(new java.awt.Font("Arial Black",0,16));
            panelFormulario.add(claveAccesoRepetirLabel, new CellConstraints(1, 4, 2, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            claveAccesoRepetirField = new JPasswordField(JPasswordField.WHEN_FOCUSED);
            claveAccesoRepetirField.setName("password");
            panelFormulario.add(claveAccesoRepetirField, new CellConstraints(2, 4, 2, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            mensajeInformativoLabel = new JLabel();
            mensajeInformativoLabel.setText("");
            mensajeInformativoLabel.setForeground(new java.awt.Color(200,0,50));
            mensajeInformativoLabel.setFont(new java.awt.Font("Arial Black",0,12));
            panelFormulario.add(mensajeInformativoLabel, new CellConstraints(2, 5, 3, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            cambiarBoton = new JButton();
            cambiarBoton.setName("Cambiar");
            cambiarBoton.setText("Cambiar");
            cambiarBoton.setActionCommand("Cambiar");
            cambiarBoton.addActionListener(adminPasswordSetupListener);
```

```
                panelFormulario.add(cambiarBoton, new CellConstraints(2, 6, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            {
                cancelarBoton = new JButton();
                cancelarBoton.setName("Cancelar");
                cancelarBoton.setText("Cancelar");
                cancelarBoton.setActionCommand("Cancelar");
                cancelarBoton.addActionListener(adminPasswordSetupListener);
                panelFormulario.add(cancelarBoton, new CellConstraints(3, 6, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            add(panelFormulario);
            setResizable(false);
            setModal(true);
            setVisible(true);
            repaint();

        }

        public JFrame getFramePadre() {
            return framePadre;
        }

        public void setFramePadre(JFrame framePadre) {
            this.framePadre = framePadre;
        }

        public JPanel getPanelFormulario() {
            return panelFormulario;
        }

        public void setPanelFormulario(JPanel panelFormulario) {
            this.panelFormulario = panelFormulario;
        }

        public JLabel getClaveAccesoLabel() {
            return claveAccesoLabel;
        }

        public void setClaveAccesoLabel(JLabel claveAccesoLabel) {
            this.claveAccesoLabel = claveAccesoLabel;
        }

        public JPasswordField getClaveAccesoField() {
            return claveAccesoField;
        }

        public void setClaveAccesoField(JPasswordField claveAccesoField) {
            this.claveAccesoField = claveAccesoField;
        }

        public JLabel getClaveAccesoRepetirLabel() {
            return claveAccesoRepetirLabel;
        }

        public void setClaveAccesoRepetirLabel(JLabel claveAccesoRepetirLabel) {
            this.claveAccesoRepetirLabel = claveAccesoRepetirLabel;
        }

        public JPasswordField getClaveAccesoRepetirField() {
            return claveAccesoRepetirField;
        }

        public void setClaveAccesoRepetirField(JPasswordField claveAccesoRepetirField) {
            this.claveAccesoRepetirField = claveAccesoRepetirField;
        }
```

```
	public JButton getCambiarBoton() {
		return cambiarBoton;
	}

	public void setCambiarBoton(JButton cambiarBoton) {
		this.cambiarBoton = cambiarBoton;
	}

	public JButton getCancelarBoton() {
		return cancelarBoton;
	}

	public void setCancelarBoton(JButton cancelarBoton) {
		this.cancelarBoton = cancelarBoton;
	}

	public JLabel getMensajeInformativoLabel() {
		return mensajeInformativoLabel;
	}

	public void setMensajeInformativoLabel(JLabel mensajeInformativoLabel) {
		this.mensajeInformativoLabel = mensajeInformativoLabel;
	}
}
```

B.6.2. hda.auth.gui.DialogMensajeTemporal.java

```
package hda.auth.gui;

import java.awt.Color;
import java.awt.Dimension;
import java.io.File;
import java.io.Serializable;
import java.util.Timer;
import java.util.TimerTask;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.CompoundBorder;

public class DialogMensajeTemporal extends JDialog implements Serializable{

	private static final long serialVersionUID = 1L;
	private JFrame framePadre;
	private JPanel panelFormulario;
	private JLabel iconoLabel;
	private JLabel mensajeLabel;
	private Timer timer;
	private String mensaje;

	{
		//Set Look & Feel
		try {
			javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
		} catch(Exception e) {
			e.printStackTrace();
		}
	}

	public DialogMensajeTemporal(JFrame framePadre, int segundos, String mensaje, int icon){
```

```
            super(framePadre,true);
            this.framePadre = framePadre;
            this.mensaje = mensaje;
            timer = new Timer();
            timer.schedule(new MensajeTemporal(this), segundos*1000);
            initComponents(icon);
      }

      private void initComponents(int icono){
            setSize(400,200);
            setLocation((framePadre.getX())+(framePadre.getWidth()/2)-(getWidth()/2) ,
(framePadre.getY())+(framePadre.getHeight()/2)-(getHeight()/2));
            setTitle("MENSAJE");
            setUndecorated(true);

            getRootPane().setOpaque (false);
            //getContentPane().setBackground (new Color (255, 255, 255, 240));
            setBackground(new Color (220, 220, 255, 230));

      JPanel unitGroup = new JPanel() {
                  private static final long serialVersionUID = 1L;
                  public Dimension getMinimumSize() {
            return getPreferredSize();
          }
          public Dimension getPreferredSize() {
            return new Dimension(400, super.getPreferredSize().height);
          }
          public Dimension getMaximumSize() {
            return getPreferredSize();
          }
      };
      unitGroup.setLayout(new BoxLayout(unitGroup, BoxLayout.LINE_AXIS));
      unitGroup.setBorder(new CompoundBorder(new
ShadowBorder(10),BorderFactory.createRaisedBevelBorder()));
      unitGroup.setOpaque(false);

      mensajeLabel = new JLabel();
            mensajeLabel.setText(mensaje);
            mensajeLabel.setForeground(new java.awt.Color(0,64,128));
            mensajeLabel.setFont(new java.awt.Font("Arial Black",0,16));
            mensajeLabel.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

            iconoLabel = new JLabel();
            iconoLabel.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));
            switch(icono){
            case 1:
                  iconoLabel.setIcon(new ImageIcon("images"+File.separator+"flecha.png"));
                  break;
            case 2:
                  iconoLabel.setIcon(new ImageIcon("images"+File.separator+"papelera.png"));
                  break;
            case 3:
                  iconoLabel.setIcon(new ImageIcon("images"+File.separator+"wait.png"));
                  break;
            }

            unitGroup.add(iconoLabel);
      unitGroup.add(mensajeLabel);

            add(unitGroup);

            setResizable(false);
            setModal(true);
            setVisible(true);
            repaint();
      }
      public JFrame getFramePadre() {
```

```
        return framePadre;
    }

    public void setFramePadre(JFrame framePadre) {
        this.framePadre = framePadre;
    }

    public JPanel getPanelFormulario() {
        return panelFormulario;
    }

    public void setPanelFormulario(JPanel panelFormulario) {
        this.panelFormulario = panelFormulario;
    }

    public JLabel getClaveAccesoLabel() {
        return mensajeLabel;
    }

    public void setClaveAccesoLabel(JLabel claveAccesoLabel) {
        this.mensajeLabel = claveAccesoLabel;
    }

    private class MensajeTemporal extends TimerTask{
        private JDialog dialog;
        public MensajeTemporal(JDialog dialog){
            this.dialog = dialog;
        }

        @Override
        public void run() {
            dialog.setModal(false);
            dialog.setVisible(false);
            dialog.dispose();
            timer.cancel();
        }

    }
    public static class TipoIcono{
        public static final int FLECHA = 1;
        public static final int PAPELERA = 2;
        public static final int WAIT = 3;
    }

}
```

B.6.3. hda.auth.gui.DialogPassword.java

```
package hda.auth.gui;


import hda.auth.gui.listeners.PasswordListener;

import java.io.Serializable;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class DialogPassword extends JDialog implements Serializable{

	private static final long serialVersionUID = 1L;
	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private JFrame framePadre;
	private JPanel panelFormulario;
	private JLabel claveAccesoLabel;
	private JPasswordField claveAccesoField;

	private JButton cambiarBoton;
	private JButton cancelarBoton;
	private JLabel mensajeInformativoLabel;

	public DialogPassword(JFrame framePadre){
		super(framePadre,true);
		this.framePadre = framePadre;
		initComponents();
	}

	private void initComponents(){
		logger.debug("iniciamos el JDialog DialogPassword");
		setSize(450,250);
		setLocation((framePadre.getX())+(framePadre.getWidth()/2)-(getWidth()/2) ,
(framePadre.getY())+(framePadre.getHeight()/2)-(getHeight()/2));
		setTitle("INTRODUZCA PASSWORD");
		PasswordListener passwordListener = new PasswordListener(this);
		addWindowListener(passwordListener);

		panelFormulario = new JPanel();
		FormLayout adminPasswordLayout = new FormLayout("max(p;140), max(p;100), max(p;100),
max(p;100)","max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30)");
		panelFormulario.setLayout(adminPasswordLayout);
		panelFormulario.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

		{
			claveAccesoLabel = new JLabel();
			claveAccesoLabel.setText("Clave acceso");
			claveAccesoLabel.setForeground(new java.awt.Color(0,64,128));
			claveAccesoLabel.setFont(new java.awt.Font("Arial Black",0,16));
			panelFormulario.add(claveAccesoLabel, new CellConstraints(1, 3, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
```

```
            }
            {
                claveAccesoField = new JPasswordField(JPasswordField.WHEN_FOCUSED);
                claveAccesoField.setName("password");
                panelFormulario.add(claveAccesoField, new CellConstraints(2, 3, 2, 1, CellConstraints.FILL,
CellConstraints.CENTER));

            }

            {
                mensajeInformativoLabel = new JLabel();
                mensajeInformativoLabel.setText("");
                mensajeInformativoLabel.setForeground(new java.awt.Color(200,0,50));
                mensajeInformativoLabel.setFont(new java.awt.Font("Arial Black",0,12));
                panelFormulario.add(mensajeInformativoLabel, new CellConstraints(2, 5, 3, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            {
                cambiarBoton = new JButton();
                cambiarBoton.setName("Entrar");
                cambiarBoton.setText("Entrar");
                cambiarBoton.setActionCommand("Entrar");
                cambiarBoton.addActionListener(passwordListener);
                panelFormulario.add(cambiarBoton, new CellConstraints(2, 6, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            {
                cancelarBoton = new JButton();
                cancelarBoton.setName("Cancelar");
                cancelarBoton.setText("Cancelar");
                cancelarBoton.setActionCommand("Cancelar");
                cancelarBoton.addActionListener(passwordListener);
                panelFormulario.add(cancelarBoton, new CellConstraints(3, 6, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            add(panelFormulario);
            setResizable(false);
            setModal(true);
            setVisible(true);
            repaint();

        }

        public JFrame getFramePadre() {
            return framePadre;
        }

        public void setFramePadre(JFrame framePadre) {
            this.framePadre = framePadre;
        }

        public JPanel getPanelFormulario() {
            return panelFormulario;
        }

        public void setPanelFormulario(JPanel panelFormulario) {
            this.panelFormulario = panelFormulario;
        }

        public JLabel getClaveAccesoLabel() {
            return claveAccesoLabel;
        }

        public void setClaveAccesoLabel(JLabel claveAccesoLabel) {
            this.claveAccesoLabel = claveAccesoLabel;
        }
```

```
	public JPasswordField getClaveAccesoField() {
		return claveAccesoField;
	}

	public void setClaveAccesoField(JPasswordField claveAccesoField) {
		this.claveAccesoField = claveAccesoField;
	}

	public JButton getCambiarBoton() {
		return cambiarBoton;
	}

	public void setCambiarBoton(JButton cambiarBoton) {
		this.cambiarBoton = cambiarBoton;
	}

	public JButton getCancelarBoton() {
		return cancelarBoton;
	}

	public void setCancelarBoton(JButton cancelarBoton) {
		this.cancelarBoton = cancelarBoton;
	}

	public JLabel getMensajeInformativoLabel() {
		return mensajeInformativoLabel;
	}

	public void setMensajeInformativoLabel(JLabel mensajeInformativoLabel) {
		this.mensajeInformativoLabel = mensajeInformativoLabel;
	}
}
```

B.6.4. hda.auth.gui.DialogRegistrarTrazo.java

```
package hda.auth.gui;

import java.io.Serializable;
import javax.swing.JDialog;
import javax.swing.JFrame;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class DialogRegistrarTrazo extends JDialog implements Serializable{

	private static final long serialVersionUID = 1L;
	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	private RegistrarTrazoPanel panelDibujo;

	public DialogRegistrarTrazo(JFrame framePrincipal){
		super(framePrincipal,true);
		initComponents();
	}

	private void initComponents(){
		logger.debug("iniciamos el JDialog RegistrarTrazo");
		setTitle("REGISTRAR TRAZO");
		setAlwaysOnTop(true);
		panelDibujo = new RegistrarTrazoPanel(this);
		add(panelDibujo);
		setSize(panelDibujo.getWidth(),panelDibujo.getHeight());
		setResizable(false);
		setVisible(true);
	}

}
```

B.6.5. hda.auth.gui.DialogValidarTrazo.java

```
package hda.auth.gui;


import java.io.Serializable;

import javax.swing.JDialog;
import javax.swing.JFrame;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class DialogValidarTrazo extends JDialog implements Serializable{

	private static final long serialVersionUID = 1L;
	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	private ValidarTrazoPanel panelDibujo;

	public DialogValidarTrazo(JFrame framePrincipal, int tipo_operacion){
		super(framePrincipal,true);
		initComponents(tipo_operacion);
	}

	private void initComponents(int tipo_operacion){
		logger.debug("iniciamos el JDialog ValidarTrazo");
		setTitle("VALIDAR TRAZO");
		setAlwaysOnTop(true);
		panelDibujo = new ValidarTrazoPanel(this,tipo_operacion);
		add(panelDibujo);
		setSize(panelDibujo.getWidth(),panelDibujo.getHeight());
		setResizable(false);
		setVisible(true);

	}
	public static class TipoOperacion{
		public static final int LOGIN = 1;
		public static final int CAMBIO_TRAZO = 2;
	}

}
```

B.6.6. hda.auth.gui.GenerarClavesPanel.java

```
package hda.auth.gui;

import java.awt.Dimension;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JPanel;

public class GenerarClavesPanel extends JPanel {

	private static final long serialVersionUID = 1L;
	JButton generarCertificadosBoton;
	JButton generarClaveAdminBoton;


   String title;

   public GenerarClavesPanel(JPanel myController, String myTitle,ActionListener listener) {
      setBorder(BorderFactory.createCompoundBorder(BorderFactory.createTitledBorder(myTitle),
BorderFactory.createEmptyBorder(5,5,5,5)));
      title = myTitle;

      JPanel unitGroup = new JPanel() {
                  private static final long serialVersionUID = 1L;
                  public Dimension getMinimumSize() {
            return getPreferredSize();
         }
         public Dimension getPreferredSize() {
            return new Dimension(300, super.getPreferredSize().height);
         }
         public Dimension getMaximumSize() {
            return getPreferredSize();
         }
      };
      unitGroup.setLayout(new BoxLayout(unitGroup, BoxLayout.LINE_AXIS));
      unitGroup.setBorder(BorderFactory.createEmptyBorder(0,0,0,5));

      generarClaveAdminBoton = new JButton();
      generarClaveAdminBoton.setName("GenerarClaveAdmin");
      generarClaveAdminBoton.setText("Generar Clave Admin");
      generarClaveAdminBoton.setActionCommand("GenerarClaveAdmin");
      generarClaveAdminBoton.addActionListener(listener);

      generarCertificadosBoton = new JButton();
      generarCertificadosBoton.setName("GenerarCertificados");
      generarCertificadosBoton.setText("Generar Certificados");
      generarCertificadosBoton.setActionCommand("GenerarCertificados");
      generarCertificadosBoton.addActionListener(listener);

      unitGroup.add(generarClaveAdminBoton);
      unitGroup.add(generarCertificadosBoton);

      add(unitGroup);

   }

}
```

B.6.7. hda.auth.gui.ParamTolerancePanel.java

```
package hda.auth.gui;

import hda.auth.config.bean.LockFactoryParameterConfigBean;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.text.NumberFormat;
import java.util.ArrayList;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.DefaultBoundedRangeModel;
import javax.swing.JComboBox;
import javax.swing.JFormattedTextField;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.text.NumberFormatter;

public class ParamTolerancePanel extends JPanel implements ActionListener, ChangeListener,
PropertyChangeListener {

    private static final long serialVersionUID = 1L;
    private ArrayList<LockFactoryParameterConfigBean> lock_params_configurations;
  JFormattedTextField textField;
  JComboBox paramSelector;
  JSlider slider;

  DefaultBoundedRangeModel sliderModel;
  String title;
  NumberFormat numberFormat;

  final static int MAX = 100;

  public ParamTolerancePanel(JPanel myController, String myTitle, ArrayList<LockFactoryParameterConfigBean>
lock_params_configurations, DefaultBoundedRangeModel myModel) {
    setBorder(BorderFactory.createCompoundBorder(BorderFactory.createTitledBorder(myTitle),
BorderFactory.createEmptyBorder(5,5,5,5)));

    //Save arguments in instance variables.
    this.lock_params_configurations = lock_params_configurations;
    title = myTitle;
    sliderModel = myModel;

    //Create the text field format, and then the text field.
    numberFormat = NumberFormat.getNumberInstance();
    numberFormat.setMaximumFractionDigits(2);
    NumberFormatter formatter = new NumberFormatter(numberFormat);
    formatter.setAllowsInvalid(false);
    formatter.setCommitsOnValidEdit(true);
    textField = new JFormattedTextField(formatter);
    textField.setColumns(5);
    textField.setValue(lock_params_configurations.get(0).getTolerance());
    textField.addPropertyChangeListener(this);


    //Add the combo box.
    paramSelector = new JComboBox();
```

```
    for (int i = 0; i < lock_params_configurations.size(); i++) { //Populate it.
       paramSelector.addItem(lock_params_configurations.get(i).getName());
    }
    paramSelector.setSelectedIndex(0);
    paramSelector.addActionListener(this);

    //Add the slider.
    slider = new JSlider(sliderModel);
    sliderModel.addChangeListener(this);
    sliderModel.setValue(lock_params_configurations.get(paramSelector.getSelectedIndex()).getTolerance());

    //Make the text field/slider group a fixed size
    //to make stacked ConversionPanels nicely aligned.
    JPanel unitGroup = new JPanel() {
                private static final long serialVersionUID = 1L;
                public Dimension getMinimumSize() {
          return getPreferredSize();
       }
       public Dimension getPreferredSize() {
          return new Dimension(300, super.getPreferredSize().height);
       }
       public Dimension getMaximumSize() {
          return getPreferredSize();
       }
    };
    unitGroup.setLayout(new BoxLayout(unitGroup, BoxLayout.PAGE_AXIS));
    unitGroup.setBorder(BorderFactory.createEmptyBorder(0,0,0,5));
    unitGroup.add(textField);
    unitGroup.add(slider);

    //Create a subpanel so the combo box isn't too tall
    //and is sufficiently wide.
    JPanel chooserPanel = new JPanel();
    chooserPanel.setLayout(new BoxLayout(chooserPanel, BoxLayout.PAGE_AXIS));

    chooserPanel.add(paramSelector);
    chooserPanel.add(Box.createHorizontalStrut(100));

    //Put everything together.
    unitGroup.setAlignmentY(TOP_ALIGNMENT);
    chooserPanel.setAlignmentY(TOP_ALIGNMENT);
    setLayout(new BoxLayout(this, BoxLayout.LINE_AXIS));
    add(unitGroup);
    add(chooserPanel);
}


/** Updates the text field when the main data model is updated. */
public void stateChanged(ChangeEvent e) {
    int min = sliderModel.getMinimum();
    int max = sliderModel.getMaximum();
    int value = sliderModel.getValue();
    NumberFormatter formatter = (NumberFormatter)textField.getFormatter();

    formatter.setMinimum(min);
    formatter.setMaximum(max);
    textField.setValue(value);
    lock_params_configurations.get(paramSelector.getSelectedIndex()).setTolerance(sliderModel.getValue());
}

/**
 * Responds to the user choosing a new unit from the combo box.
 */
public void actionPerformed(ActionEvent e) {
    int i = paramSelector.getSelectedIndex();
    sliderModel.setValue(lock_params_configurations.get(i).getTolerance());
}
```

```
    /**
     * Detects when the value of the text field (not necessarily the same
     * number as you'd get from getText) changes.
     */
    public void propertyChange(PropertyChangeEvent e) {
        if ("value".equals(e.getPropertyName())) {
            Number value = (Number)e.getNewValue();
            sliderModel.setValue(value.intValue());
        }
    }
}
```

B.6.8. hda.auth.gui.PrincipalGUI.java

```
package hda.auth.gui;

import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.config.bean.KeyFactoryParameterConfigBean;
import hda.auth.config.bean.LockFactoryParameterConfigBean;
import hda.auth.gui.listeners.PrincipalGUIListener;
import hda.auth.keys.Key;
import hda.auth.keys.KeyFactory;
import hda.auth.keys.KeyFactoryConfigException;
import hda.auth.locks.Lock;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.FlowLayout;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.security.Security;

import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class PrincipalGUI extends javax.swing.JFrame {
	private static final long serialVersionUID = 1L;
	private static Logger logger = null;
	private static PrincipalGUI inst;
	private Lock lock;
	private Key key;

	//Paneles
	private JPanel panelPrincipal;
	private JTabbedPane tabs;
	private TabLoginPanel tabLoginPanel;
	private TabSetupPanel tabSetupPanel;

	//Pestañas
	private JPanel tabLogin;
	private JPanel tabSetup;

	//Cabecera
	private JLabel iconoLabel;
	private JPanel TextoCabeceraPanel;
	private JLabel SubCabeceraLabel;
	private JLabel CabeceraLabel;
	private JPanel CabeceraPanel;

	{
		//Set Look & Feel
		try {
			javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
		} catch(Exception e) {
			logger.error(e.getMessage());
```

```
        }
    }

    public Lock getLock() {
        return lock;
    }

    public void setLock(Lock lock) {
        this.lock = lock;
    }

    public Key getKey() {
        return key;
    }

    public void setKey(Key key) {
        this.key = key;
    }

    public static void main(String[] args) {
        try {
            //leemos la configuracion principal
            Config.getInstance();
            //configuramos el Logger
            System.setProperty("log4j.configurationFile",
((Config.getInstance()).getModuleConfigByName("Logger")).getConfigFile());

            logger = LogManager.getLogger(PrincipalGUI.class.getName());
            logger.info("Leida configuracion principal <"+System.getProperty("mainconfig")+">");
            logger.debug("Fichero Propiedades Logger <"+System.getProperty("log4j.configurationFile")+">");

            //LockFatory
            LockFactory.getInstance();
            logger.info("LockFactory configurado");
            logger.debug("LockFactory definido con
"+LockFactory.getInstance().getLock_params_configurations().size()+" parametros");
            logger.debug("LockFactory definido con "+LockFactory.getInstance().getNumero_trazos()+" trazos");
            logger.debug("LockFactory definido con dispositivo "+LockFactory.getInstance().getDispositivo());
            for(LockFactoryParameterConfigBean lockParam :
LockFactory.getInstance().getLock_params_configurations()){
                logger.debug("-------Lock Parameter------");
                logger.debug("- nombre: "+lockParam.getName());
                logger.debug("- clase: "+lockParam.getClassName());
                logger.debug("- tolerancia: "+lockParam.getTolerance());
                logger.debug("- enabled: "+lockParam.isEnabled());
            }

            //KeyFactory
            KeyFactory.getInstance();
            logger.info("KeyFactory configurado");
            logger.debug("KeyFactory definido con
"+KeyFactory.getInstance().getKey_params_configurations().size()+" parametros");
            logger.debug("KeyFactory definido con "+KeyFactory.getInstance().getNumero_trazos()+" trazos");
            for(KeyFactoryParameterConfigBean keyParam :
KeyFactory.getInstance().getKey_params_configurations()){
                logger.debug("-------Key Parameter------");
                logger.debug("- nombre: "+keyParam.getName());
                logger.debug("- clase: "+keyParam.getClassName());
                logger.debug("- enabled: "+keyParam.isEnabled());
            }



            Security.addProvider(new BouncyCastleProvider());
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    inst = new PrincipalGUI();
```

```
                        inst.setLocationRelativeTo(null);
                        inst.setVisible(true);
                    }
                });

        } catch (ConfigException e) {
            logger.error(e.getMessage());
            System.exit(ERROR);
        } catch (LockFactoryConfigException e) {
            logger.error(e.getMessage());
            System.exit(ERROR);
        } catch (KeyFactoryConfigException e) {
            logger.error(e.getMessage());
            System.exit(ERROR);
        }

    }

    public PrincipalGUI() {
        initGUI();
    }
    public static PrincipalGUI getInstance(){
        return inst;
    }
    public void initGUI() {
        try {
            FlowLayout thisLayout = new FlowLayout();
            getContentPane().setLayout(thisLayout);
            setResizable(true);
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            {
                panelPrincipal = new JPanel();
                getContentPane().add(panelPrincipal);
                panelPrincipal.setPreferredSize(new java.awt.Dimension(600, 600));
                {
                    CabeceraPanel = new JPanel();
                    panelPrincipal.add(CabeceraPanel);
                    CabeceraPanel.setPreferredSize(new java.awt.Dimension(600, 100));
                    CabeceraPanel.setSize(600, 100);
                    {
                        TextoCabeceraPanel = new JPanel();
                        CabeceraPanel.add(TextoCabeceraPanel);
                        TextoCabeceraPanel.setPreferredSize(new java.awt.Dimension(400, 84));
                        {
                            CabeceraLabel = new JLabel();
                            TextoCabeceraPanel.add(CabeceraLabel);
                            CabeceraLabel.setText("Sistema de Autenticación - HDA -");
                            CabeceraLabel.setForeground(new java.awt.Color(0,64,128));
                            CabeceraLabel.setFont(new java.awt.Font("Arial Black",0,16));
                            CabeceraLabel.setPreferredSize(new java.awt.Dimension(300, 35));
                        }
                        {
                            SubCabeceraLabel = new JLabel();
                            TextoCabeceraPanel.add(SubCabeceraLabel);
                            SubCabeceraLabel.setText("PFG: Gonzalo Anton Garcia");
                            SubCabeceraLabel.setForeground(new java.awt.Color(0,64,128));
                            SubCabeceraLabel.setFont(new java.awt.Font("Arial Black",0,12));
                            SubCabeceraLabel.setPreferredSize(new java.awt.Dimension(300, 35));
                        }
                    }
                    {
                        iconoLabel = new JLabel();
                        CabeceraPanel.add(iconoLabel);
                        iconoLabel.setIcon(new ImageIcon("images"+File.separator+"logo.png"));
                        iconoLabel.setPreferredSize(new java.awt.Dimension(150, 100));
                        iconoLabel.setOpaque(true);
                    }
```

```
                    }

                    tabs = new JTabbedPane();
                    panelPrincipal.add(tabs);
                    tabs.setPreferredSize(new java.awt.Dimension(600, 600));
                    tabs.addChangeListener(new PrincipalGUIListener(this));
                    tabs.setName("tabs");

                    {
                        //PESTAÑA LOGIN
                        tabLoginPanel = new TabLoginPanel();
                        tabs.addTab("1. Principal", null, tabLoginPanel, null);

                        //PESTAÑA SETUP
                        tabSetupPanel = new TabSetupPanel(this);
                        tabs.addTab("2. Configurar", null, tabSetupPanel, null);

                    }
                }
                tabLoginPanel.getValidarTrazoBoton().setEnabled(false);
                //leemos anteriores lock de fichero persistencia datos
                try {
                    File lockFile = new
File(Config.getInstance().getModuleConfigByName("LockObject").getConfigFile());
                    FileInputStream fis = new FileInputStream(lockFile);
                    ObjectInputStream ois = new ObjectInputStream(fis);
                    Object objeto = ois.readObject();
                    ois.close();
                    if(objeto instanceof Lock){
                        setLock((Lock)objeto);
                        tabLoginPanel.getValidarTrazoBoton().setEnabled(true);
                    }else{
                        tabLoginPanel.getValidarTrazoBoton().setEnabled(false);
                    }


                } catch (ConfigException e) {
                    logger.error(e.getMessage());
                    tabLoginPanel.getValidarTrazoBoton().setEnabled(false);
                } catch (FileNotFoundException e) {
                    logger.error(e.getMessage());
                    tabLoginPanel.getValidarTrazoBoton().setEnabled(false);
                } catch (IOException e) {
                    logger.error(e.getMessage());
                    tabLoginPanel.getValidarTrazoBoton().setEnabled(false);
                }
                pack();
                inst = this;
            } catch (Exception e) {
                logger.error(e.getMessage());
            }
        }

        public JPanel getPanelPrincipal() {
            return panelPrincipal;
        }

        public void setPanelPrincipal(JPanel panelPrincipal) {
            this.panelPrincipal = panelPrincipal;
        }

        public JTabbedPane getTabs() {
            return tabs;
        }

        public void setTabs(JTabbedPane tabs) {
            this.tabs = tabs;
```

```
    }

    public JPanel getTabLogin() {
        return tabLogin;
    }

    public void setTabLogin(JPanel tabLogin) {
        this.tabLogin = tabLogin;
    }

    public static Logger getLogger() {
        return logger;
    }

    public static void setLogger(Logger logger) {
        PrincipalGUI.logger = logger;
    }

    public JPanel getTabSetup() {
        return tabSetup;
    }

    public void setTabSetup(JPanel tabSetup) {
        this.tabSetup = tabSetup;
    }

    public JLabel getIconoLabel() {
        return iconoLabel;
    }

    public void setIconoLabel(JLabel iconoLabel) {
        this.iconoLabel = iconoLabel;
    }

    public JPanel getTextoCabeceraPanel() {
        return TextoCabeceraPanel;
    }

    public void setTextoCabeceraPanel(JPanel textoCabeceraPanel) {
        TextoCabeceraPanel = textoCabeceraPanel;
    }

    public JLabel getSubCabeceraLabel() {
        return SubCabeceraLabel;
    }

    public void setSubCabeceraLabel(JLabel subCabeceraLabel) {
        SubCabeceraLabel = subCabeceraLabel;
    }

    public JLabel getCabeceraLabel() {
        return CabeceraLabel;
    }

    public void setCabeceraLabel(JLabel cabeceraLabel) {
        CabeceraLabel = cabeceraLabel;
    }

    public JPanel getCabeceraPanel() {
        return CabeceraPanel;
    }

    public void setCabeceraPanel(JPanel cabeceraPanel) {
        CabeceraPanel = cabeceraPanel;
    }

    public TabLoginPanel getTabLoginPanel() {
```

```
        return tabLoginPanel;
    }

    public void setTabLoginPanel(TabLoginPanel tabLoginPanel) {
        this.tabLoginPanel = tabLoginPanel;
    }

    public TabSetupPanel getTabSetupPanel() {
        return tabSetupPanel;
    }

    public void setTabSetupPanel(TabSetupPanel tabSetupPanel) {
        this.tabSetupPanel = tabSetupPanel;
    }

}
```

B.6.9. hda.auth.gui.RegistrarTrazoPanel.java

```
package hda.auth.gui;

import hda.auth.gui.listeners.RegistrarTrazoListener;
import hda.auth.img.Trazo;

import java.awt.AWTException;
import java.awt.GraphicsConfiguration;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Robot;
import java.io.Serializable;
import java.util.ArrayList;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JPanel;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class RegistrarTrazoPanel extends JPanel implements Serializable {
	private static final long serialVersionUID = 1L;
	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	private JDialog dialog;
	private Point inicioArrastre;
	private Point finArrastre;
	private ArrayList<Trazo> trazos = new ArrayList<Trazo>();
	private GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
	private GraphicsDevice[] gs = ge.getScreenDevices();

	/**
	 * Guarda una instancia del {@link JFrame} que lo contiene
	 */
	public RegistrarTrazoPanel(JDialog dialog){
		this.setDialog(dialog);
		inicioArrastre = new Point(-1,-1);
		finArrastre = new Point(-1,-1);
		initComponents();
	}

	/**
	 * @return Devuelve un objeto de tipo {@link Point} que representa el inicio del arrastre del Mouse
	 */
	public Point getInicioArrastre() {
		return inicioArrastre;
	}

	public void setInicioArrastre(Point inicioArrastre) {
		this.inicioArrastre = inicioArrastre;
	}

	public Point getFinArrastre() {
		return finArrastre;
	}

	public void setFinArrastre(Point finArrastre) {
		this.finArrastre = finArrastre;
	}

	public ArrayList<Trazo> getTrazos() {
		return trazos;
```

```
    }

    public void setTrazos(ArrayList<Trazo> trazos) {
        this.trazos = trazos;
    }

    public void setDialog(JDialog dialog) {
        this.dialog = dialog;
    }

    public JDialog getDialog() {
        return dialog;
    }
    public Trazo getTrazoVacio(){
        Trazo trazo = new Trazo(getWidth(),getHeight());
        trazos.add(trazo);
        return trazo;
    }

    private void initComponents(){
        logger.debug("iniciamos Panel RegistrarTrazo");
        int xoffs = 0;
        int yoffs = 0;
        GraphicsDevice gd = null;

        gd = gs[0];
        GraphicsConfiguration[] gc = gd.getConfigurations();
        Rectangle gcBounds = gc[0].getBounds();
        xoffs = gcBounds.x;
        yoffs = gcBounds.y;

        setFocusable(true);
        setLocation(xoffs, yoffs);
        setSize(gc[0].getBounds().width, gc[0].getBounds().height);
        this.setLocation(0, 0);


        //Creamos y añadimos el listener de Eventos del ratón
        RegistrarTrazoListener rtmel = new RegistrarTrazoListener(this);
        addMouseMotionListener(rtmel);//movimiento
        addMouseListener(rtmel);//botones

        try {
            Robot robot = new Robot();
            robot.mouseMove(getWidth()/2, getHeight()/2);
        } catch (AWTException e) {
            e.printStackTrace();
        }

    }


}
```

B.6.10. hda.auth.gui.SeleccionDispositivoPanel.java

```
package hda.auth.gui;

import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.Dimension;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.border.EmptyBorder;

public class SeleccionDispositivoPanel extends JPanel {

	private static final long serialVersionUID = 1L;
	JRadioButton udrawBoton;
	JRadioButton chronosTIBoton;
	JLabel udrawLabel;
	JLabel chronosLabel;


   String title;

   public SeleccionDispositivoPanel(JPanel myController, String myTitle,ActionListener listener) {
      String dispositivo;
      try {
                dispositivo = LockFactory.getInstance().getDispositivo();
            } catch (LockFactoryConfigException e) {
                dispositivo = "udraw";
            }


      setBorder(BorderFactory.createCompoundBorder(BorderFactory.createTitledBorder(myTitle),
BorderFactory.createEmptyBorder(5,5,5,5)));
      title = myTitle;

      JPanel unitGroup = new JPanel() {
                private static final long serialVersionUID = 1L;
                public Dimension getMinimumSize() {
            return getPreferredSize();
         }
         public Dimension getPreferredSize() {
            return new Dimension(300, super.getPreferredSize().height);
         }
         public Dimension getMaximumSize() {
            return getPreferredSize();
         }
      };
      unitGroup.setLayout(new BoxLayout(unitGroup, BoxLayout.X_AXIS));
      unitGroup.setBorder(BorderFactory.createEmptyBorder(0,0,0,5));

      udrawBoton = new JRadioButton();
      udrawBoton.setActionCommand("udrawDevice");
      if("udraw".equals(dispositivo)){
            udrawBoton.setSelected(true);
      }
      udrawBoton.addActionListener(listener);

      udrawLabel = new JLabel();
```

```
        udrawLabel.setText("UDraw para PS3");
        udrawLabel.setBorder(new EmptyBorder(0, 0, 0, 50));

        chronosTIBoton = new JRadioButton();
        chronosTIBoton.setActionCommand("chronosDevice");
        if("chronosTI".equals(dispositivo)){
              chronosTIBoton.setSelected(true);
        }
        chronosTIBoton.addActionListener(listener);

        chronosLabel = new JLabel();
        chronosLabel.setText("Chronos TI");
        chronosLabel.setBorder(new EmptyBorder(0, 0, 0, 50));

        ButtonGroup group = new ButtonGroup();
        group.add(udrawBoton);
        group.add(chronosTIBoton);

        unitGroup.add(udrawBoton);
        unitGroup.add(udrawLabel);
        unitGroup.add(chronosTIBoton);
        unitGroup.add(chronosLabel);


        add(unitGroup);

    }

}
```

B.6.11.hda.auth.gui.SeleccionHorasValidezPanel.java

```
package hda.auth.gui;

import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.Dimension;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.border.EmptyBorder;

public class SeleccionHorasValidezPanel extends JPanel {

        private static final long serialVersionUID = 1L;
        JRadioButton b1Boton;
        JRadioButton b2Boton;
        JRadioButton b3Boton;
        JRadioButton b4Boton;
        JRadioButton b5Boton;
        JLabel b1Label;
        JLabel b2Label;
        JLabel b3Label;
        JLabel b4Label;
        JLabel b5Label;


    String title;

    public SeleccionHorasValidezPanel(JPanel myController, String myTitle,ActionListener listener) {
```

```
	int horasvalidez;
	try {
		horasvalidez = LockFactory.getInstance().getHoras_validez();
		} catch (LockFactoryConfigException e) {
			horasvalidez = 24;
		}


	setBorder(BorderFactory.createCompoundBorder(BorderFactory.createTitledBorder(myTitle),
BorderFactory.createEmptyBorder(5,5,5,5)));
	title = myTitle;

	JPanel unitGroup = new JPanel() {
			private static final long serialVersionUID = 1L;
			public Dimension getMinimumSize() {
		return getPreferredSize();
	    }
	    public Dimension getPreferredSize() {
		return new Dimension(300, super.getPreferredSize().height);
	    }
	    public Dimension getMaximumSize() {
		return getPreferredSize();
	    }
	};
	unitGroup.setLayout(new BoxLayout(unitGroup, BoxLayout.X_AXIS));
	unitGroup.setBorder(BorderFactory.createEmptyBorder(0,0,0,5));

	b1Boton = new JRadioButton();
	b1Boton.setActionCommand("horasvalidez1");
	if(horasvalidez==1){
		b1Boton.setSelected(true);
	}
	b1Boton.addActionListener(listener);

	b1Label = new JLabel();
	b1Label.setText("1h");
	b1Label.setBorder(new EmptyBorder(0, 0, 0, 30));


	b2Boton = new JRadioButton();
	b2Boton.setActionCommand("horasvalidez2");
	if(horasvalidez==6){
		b2Boton.setSelected(true);
	}
	b2Boton.addActionListener(listener);

	b2Label = new JLabel();
	b2Label.setText("6h");
	b2Label.setBorder(new EmptyBorder(0, 0, 0, 30));

	b3Boton = new JRadioButton();
	b3Boton.setActionCommand("horasvalidez3");
	if(horasvalidez==12){
		b3Boton.setSelected(true);
	}
	b3Boton.addActionListener(listener);

	b3Label = new JLabel();
	b3Label.setText("12h");
	b3Label.setBorder(new EmptyBorder(0, 0, 0, 30));

	b4Boton = new JRadioButton();
	b4Boton.setActionCommand("horasvalidez4");
	if(horasvalidez==24){
		b4Boton.setSelected(true);
	}
	b4Boton.addActionListener(listener);
```

```
        b4Label = new JLabel();
        b4Label.setText("24h");
        b4Label.setBorder(new EmptyBorder(0, 0, 0, 30));

        b5Boton = new JRadioButton();
        b5Boton.setActionCommand("horasvalidez5");
        if(horasvalidez==48){
            b5Boton.setSelected(true);
        }
        b5Boton.addActionListener(listener);

        b5Label = new JLabel();
        b5Label.setText("48h");
        b5Label.setBorder(new EmptyBorder(0, 0, 0, 30));

        ButtonGroup group = new ButtonGroup();
        group.add(b1Boton);
        group.add(b2Boton);
        group.add(b3Boton);
        group.add(b4Boton);
        group.add(b5Boton);

        unitGroup.add(b1Boton);
        unitGroup.add(b1Label);
        unitGroup.add(b2Boton);
        unitGroup.add(b2Label);
        unitGroup.add(b3Boton);
        unitGroup.add(b3Label);
        unitGroup.add(b4Boton);
        unitGroup.add(b4Label);
        unitGroup.add(b5Boton);
        unitGroup.add(b5Label);


        add(unitGroup);

    }

}
```

B.6.12.hda.auth.gui.ShadowBorder.java

```
package hda.auth.gui;

import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Insets;
import java.awt.Rectangle;
import java.awt.geom.Area;
import java.awt.image.BufferedImage;
import javax.swing.border.Border;

import com.jhlabs.image.GaussianFilter;

public class ShadowBorder implements Border {

  private static final Color SHADOW_COLOR = Color.BLACK;
  private static final float SHADOW_ALPHA = 0.8f;

  private int size;
  public ShadowBorder(int size) {
```

```
    if( size < 0 )
      throw new IllegalArgumentException();

    this.size = size;
  }

  @Override
  public void paintBorder(Component c, Graphics g, int x, int y, int width, int height) {

    int widthFix = width - (size * 2);
    int heightFix = height - (size * 2);


    BufferedImage shadow = new BufferedImage(size * 2 + width, size * 2 + height,
BufferedImage.TYPE_INT_ARGB);
    Graphics2D imgGraphics = shadow.createGraphics();
    imgGraphics.setColor(SHADOW_COLOR);

    imgGraphics.fillRect(size, size, widthFix, heightFix);

    imgGraphics.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_IN, SHADOW_ALPHA));
    imgGraphics.fillRect(0, 0, shadow.getWidth(), shadow.getHeight());
    imgGraphics.dispose();

    GaussianFilter filter = new GaussianFilter(size * 2);
    shadow = filter.filter(shadow, null);

    Area shadowArea = new Area(new Rectangle(0, 0, width, height));
    shadowArea.subtract(new Area(new Rectangle(0, 0, widthFix, heightFix)));

    Graphics2D gg = (Graphics2D)g.create();
    gg.setClip(shadowArea);
    gg.drawImage(shadow, 0, 0, c);
    gg.dispose();
  }

  @Override
  public Insets getBorderInsets(Component c) {

    return new Insets(0, 0, size * 2, size * 2);
  }

  @Override
  public boolean isBorderOpaque() {

    return false;
  }
}
```

B.6.13. hda.auth.gui.TabLoginPanel.java

```
package hda.auth.gui;



import hda.auth.gui.listeners.TabLoginListener;

import java.io.File;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class TabLoginPanel extends JPanel {

	private static final long serialVersionUID = 1L;

	TabLoginListener tabLoginListener = new TabLoginListener(this);
	//Etiquetas Botones y Resultado principales Simetrico
	private JLabel validarTrazoLabel;
	private JButton validarTrazoBoton;
	private JLabel validarTrazoResLabel;
	private JLabel registrarTrazoLabel;
	private JButton registrarTrazoBoton;
	private JLabel registrarTrazoResLabel;
	private JLabel logoutLabel;
	private JButton logoutBoton;
	private JLabel logoutResLabel;

	public TabLoginPanel(){
		initGUI();
	}
	public void initGUI(){
		removeAll();
		setName("loginPanel");
		FormLayout loginPanelLayout = new
FormLayout("max(p;10),max(p;140),max(p;140),max(p;140),max(p;140),max(p;10)",    "max(p;15), max(p;15),
max(p;15), max(p;15), max(p;15), max(p;15), max(p;15), max(p;15), max(p;15), max(p;15), max(p;15),
max(p;15)");
		setLayout(loginPanelLayout);
		setPreferredSize(new java.awt.Dimension(600, 600));
		{
			validarTrazoLabel = new JLabel();
			validarTrazoLabel.setText("1.- Validar");
			validarTrazoLabel.setForeground(new java.awt.Color(0,64,128));
			validarTrazoLabel.setFont(new java.awt.Font("Arial Black",0,16));
			add(validarTrazoLabel, new CellConstraints("2, 7, 1, 1, default, default"));

		}
		{
			validarTrazoBoton = new JButton();
			validarTrazoBoton.setIcon(new ImageIcon("images"+File.separator+"validar.png"));
			validarTrazoBoton.setName("validarTrazo");
			validarTrazoBoton.setActionCommand("validarTrazo");
			validarTrazoBoton.addActionListener(tabLoginListener);
			add(validarTrazoBoton, new CellConstraints("3, 7, 2, 1, fill, default"));
			validarTrazoBoton.setEnabled(false);


		}
```

```
        {
            validarTrazoResLabel = new JLabel();
            validarTrazoResLabel.setIcon(new ImageIcon("images"+File.separator+"ok.png"));
            add(validarTrazoResLabel, new CellConstraints("5, 7, 1, 1, left, default"));
            validarTrazoResLabel.setVisible(false);
        }
        {
            registrarTrazoLabel = new JLabel();
            registrarTrazoLabel.setText("2.- Registrar");
            registrarTrazoLabel.setForeground(new java.awt.Color(0,64,128));
            registrarTrazoLabel.setFont(new java.awt.Font("Arial Black",0,16));
            add(registrarTrazoLabel, new CellConstraints("2, 8, 1, 1, default, default"));

        }
        {
            registrarTrazoBoton = new JButton();
            registrarTrazoBoton.setIcon(new ImageIcon("images"+File.separator+"registrar.png"));
            registrarTrazoBoton.setName("registrarTrazo");
            registrarTrazoBoton.setActionCommand("registrarTrazo");
            registrarTrazoBoton.addActionListener(tabLoginListener);
            add(registrarTrazoBoton, new CellConstraints("3, 8, 2, 1, default, center"));

        }
        {
            registrarTrazoResLabel = new JLabel();
            registrarTrazoResLabel.setIcon(new ImageIcon("images"+File.separator+"ok.png"));
            add(registrarTrazoResLabel, new CellConstraints("5, 8, 1, 1, left, default"));
            registrarTrazoResLabel.setVisible(false);

        }
        {
            logoutLabel = new JLabel();
            add(logoutLabel, new CellConstraints("2, 9, 1, 1, default, default"));
            logoutLabel.setText("3.- Logout");
            logoutLabel.setForeground(new java.awt.Color(0,64,128));
            logoutLabel.setFont(new java.awt.Font("Arial Black",0,16));

        }
        {
            logoutBoton = new JButton();
            logoutBoton.setIcon(new ImageIcon("images"+File.separator+"salir.png"));
            logoutBoton.setName("logout");
            logoutBoton.setActionCommand("logout");
            logoutBoton.addActionListener(tabLoginListener);
            add(logoutBoton, new CellConstraints("3, 9, 2, 1, default, center"));

        }
        {
            logoutResLabel = new JLabel();
            logoutResLabel.setIcon(new ImageIcon("images"+File.separator+"ok.png"));
            add(logoutResLabel, new CellConstraints("5, 9, 1, 1, left, default"));
            logoutResLabel.setVisible(false);

        }
        addMouseListener(tabLoginListener);
        repaint();

    }

    public void limpiar(){
        //limpiamos toda la plantilla
        if(validarTrazoResLabel!=null)
            validarTrazoResLabel.setVisible(false);
        if(validarTrazoResLabel!=null)
            validarTrazoResLabel.setVisible(false);
        if(registrarTrazoResLabel!=null)
            registrarTrazoResLabel.setVisible(false);
```

```
        if(logoutResLabel!=null)
            logoutResLabel.setVisible(false);

    }

    public JLabel getValidarTrazoLabel() {
        return validarTrazoLabel;
    }

    public void setValidarTrazoLabel(JLabel validarTrazoLabel) {
        this.validarTrazoLabel = validarTrazoLabel;
    }

    public JButton getValidarTrazoBoton() {
        return validarTrazoBoton;
    }

    public void setValidarTrazoBoton(JButton validarTrazoBoton) {
        this.validarTrazoBoton = validarTrazoBoton;
    }

    public JLabel getValidarTrazoResLabel() {
        return validarTrazoResLabel;
    }

    public void setValidarTrazoResLabel(JLabel validarTrazoResLabel) {
        this.validarTrazoResLabel = validarTrazoResLabel;
    }

    public JLabel getRegistrarTrazoLabel() {
        return registrarTrazoLabel;
    }

    public void setRegistrarTrazoLabel(JLabel registrarTrazoLabel) {
        this.registrarTrazoLabel = registrarTrazoLabel;
    }

    public JButton getRegistrarTrazoBoton() {
        return registrarTrazoBoton;
    }

    public void setRegistrarTrazoBoton(JButton registrarTrazoBoton) {
        this.registrarTrazoBoton = registrarTrazoBoton;
    }

    public JLabel getRegistrarTrazoResLabel() {
        return registrarTrazoResLabel;
    }

    public void setRegistrarTrazoResLabel(JLabel registrarTrazoResLabel) {
        this.registrarTrazoResLabel = registrarTrazoResLabel;
    }

    public JLabel getLogoutLabel() {
        return logoutLabel;
    }

    public void setLogoutLabel(JLabel logoutLabel) {
        this.logoutLabel = logoutLabel;
    }

    public JButton getLogoutBoton() {
        return logoutBoton;
    }

    public void setLogoutBoton(JButton logoutBoton) {
        this.logoutBoton = logoutBoton;
```

```
    }

    public JLabel getLogoutResLabel() {
        return logoutResLabel;
    }

    public void setLogoutResLabel(JLabel logoutResLabel) {
        this.logoutResLabel = logoutResLabel;
    }

}
```

B.6.14. hda.auth.gui.TabSetupPanel.java

```
package hda.auth.gui;

import hda.auth.gui.listeners.TabSetupListener;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.io.File;

import javax.swing.DefaultBoundedRangeModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class TabSetupPanel extends JPanel{

	private static final long serialVersionUID = 1L;
	private TabSetupListener tabSetupListener = new TabSetupListener(this);
	private boolean isAdminLogged = false;
	private ParamTolerancePanel param_tolerance_panel;


	private JLabel mensajeInformativoLabel;
	private JLabel mensajeInformativoIconoLabel;

	private JLabel claveAccesoLabel;
	private JLabel claveAccesoResLabel;
	private JPasswordField claveAccesoField;
	private JButton claveAccesoBoton;
	private JButton guardarConfigBoton;

	private JButton finalizarConfigBoton;

	private JFrame framePrincipal;

	public TabSetupPanel(JFrame framePrincipal) throws LockFactoryConfigException{
		this.framePrincipal = framePrincipal;
		initGUI();
	}

	public void initGUI() throws LockFactoryConfigException{
		removeAll();
		setName("setupPanel");
		FormLayout setupPanelLayout = new FormLayout("max(p;10), max(p;140), max(p;140), max(p;140),
max(p;140), max(p;10)","max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30),
max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30)");
		setLayout(setupPanelLayout);
		setPreferredSize(new java.awt.Dimension(600, 400));
		setOpaque(false);

		//menu de configuracion de administradores
		if(isAdminLogged){
			{
				DefaultBoundedRangeModel model = new DefaultBoundedRangeModel();
				param_tolerance_panel = new ParamTolerancePanel(this,"Tolerancia
Base",LockFactory.getInstance().getLock_params_configurations(),model);
				add(param_tolerance_panel, new CellConstraints(2, 1, 5, 1, CellConstraints.DEFAULT,
CellConstraints.DEFAULT));
```

```
                    param_tolerance_panel.setForeground(new java.awt.Color(0,64,128));
                    param_tolerance_panel.setFont(new java.awt.Font("Arial Black",0,16));
                }
                {
                    SeleccionDispositivoPanel seleccionDispositivoPanel = new
SeleccionDispositivoPanel(this,"Seleccion Dispositivo", tabSetupListener);
                    add(seleccionDispositivoPanel, new CellConstraints(2, 2, 5, 1, CellConstraints.DEFAULT,
CellConstraints.DEFAULT));
                    seleccionDispositivoPanel.setForeground(new java.awt.Color(0,64,128));
                    seleccionDispositivoPanel.setFont(new java.awt.Font("Arial Black",0,16));
                }
                {
                    SeleccionHorasValidezPanel seleccionHorasValidezPanel = new
SeleccionHorasValidezPanel(this,"Seleccion Horas Validez", tabSetupListener);
                    add(seleccionHorasValidezPanel, new CellConstraints(2, 3, 5, 1, CellConstraints.DEFAULT,
CellConstraints.DEFAULT));
                    seleccionHorasValidezPanel.setForeground(new java.awt.Color(0,64,128));
                    seleccionHorasValidezPanel.setFont(new java.awt.Font("Arial Black",0,16));
                }
                {
                    GenerarClavesPanel generarClavesPanel = new GenerarClavesPanel(this,"Generar Claves",
tabSetupListener);
                    add(generarClavesPanel, new CellConstraints(2, 4, 5, 1, CellConstraints.DEFAULT,
CellConstraints.DEFAULT));
                    generarClavesPanel.setForeground(new java.awt.Color(0,64,128));
                    generarClavesPanel.setFont(new java.awt.Font("Arial Black",0,16));
                }

                //MensajeInformativo
                {
                    mensajeInformativoIconoLabel = new JLabel();
                    add(mensajeInformativoIconoLabel, new CellConstraints(2, 5, 1, 1, CellConstraints.RIGHT,
CellConstraints.DEFAULT));
                    mensajeInformativoIconoLabel.setIcon(new ImageIcon("images"+File.separator+"ok.png"));
                    mensajeInformativoIconoLabel.setVisible(false);
                }

                {
                    mensajeInformativoLabel = new JLabel();
                    add(mensajeInformativoLabel, new CellConstraints(3, 5, 2, 1, CellConstraints.CENTER,
CellConstraints.DEFAULT));
                    mensajeInformativoLabel.setText("");
                    mensajeInformativoLabel.setFont(new java.awt.Font("Arial Black",0,14));
                    mensajeInformativoLabel.setVisible(false);
                }

                {
                    guardarConfigBoton = new JButton();
                    add(guardarConfigBoton, new CellConstraints("3, 6, 1, 1, default, center"));
                    guardarConfigBoton.setName("Guardar");
                    guardarConfigBoton.setActionCommand("Guardar");
                    guardarConfigBoton.setText("Guardar");
                    guardarConfigBoton.addActionListener(tabSetupListener);
                }
                {
                    finalizarConfigBoton = new JButton();
                    add(finalizarConfigBoton, new CellConstraints("4, 6, 1, 1, default, center"));
                    finalizarConfigBoton.setName("Salir Configuración");
                    finalizarConfigBoton.setActionCommand("SalirConfig");
                    finalizarConfigBoton.setText("Salir Configuración");
                    finalizarConfigBoton.addActionListener(tabSetupListener);
                }
                //
            }else{
                {
                    claveAccesoLabel = new JLabel();
                    claveAccesoLabel.setText("Clave Acceso");
```

```
                claveAccesoLabel.setForeground(new java.awt.Color(0,64,128));
                claveAccesoLabel.setFont(new java.awt.Font("Arial Black",0,16));
                add(claveAccesoLabel, new CellConstraints("2, 4, 1, 1, default, default"));
            }
            {
                claveAccesoField = new JPasswordField(JPasswordField.WHEN_FOCUSED);
                claveAccesoField.setName("password");
                add(claveAccesoField, new CellConstraints("3, 4, 2, 1, fill, default"));

            }
            {
                claveAccesoBoton = new JButton();
                claveAccesoBoton.setName("Entrar");
                claveAccesoBoton.setText("Entrar");
                claveAccesoBoton.setActionCommand("Entrar");
                claveAccesoBoton.addActionListener(tabSetupListener);
                add(claveAccesoBoton, new CellConstraints("5, 4, 1, 1, fill, default"));

            }
            {
                claveAccesoResLabel = new JLabel();
                claveAccesoResLabel.setForeground(new java.awt.Color(200,30,30));
                claveAccesoResLabel.setFont(new java.awt.Font("Arial Black",0,12));
                add(claveAccesoResLabel, new CellConstraints("3, 6, 2, 1, default, default"));
                claveAccesoResLabel.setText("");
                claveAccesoResLabel.setVisible(false);
            }
        }
        repaint();
    }

    public void limpiar(){

    }


    public JLabel getMensajeInformativoLabel() {
        return mensajeInformativoLabel;
    }

    public void setMensajeInformativoLabel(
            JLabel generarClaveAsimetricaResLabel) {
        this.mensajeInformativoLabel = generarClaveAsimetricaResLabel;
    }

    public JPasswordField getClaveAccesoField() {
        return claveAccesoField;
    }

    public void setClaveAccesoField(JPasswordField claveAccesoField) {
        this.claveAccesoField = claveAccesoField;
    }

    public boolean isAdminLogged() {
        return isAdminLogged;
    }

    public void setAdminLogged(boolean isAdminLogged) {
        this.isAdminLogged = isAdminLogged;
    }

    public JLabel getClaveAccesoResLabel() {
        return claveAccesoResLabel;
    }

    public void setClaveAccesoResLabel(JLabel claveAccesoResLabel) {
        this.claveAccesoResLabel = claveAccesoResLabel;
```

```
    }

    public JLabel getMensajeInformativoIconoLabel() {
        return mensajeInformativoIconoLabel;
    }

    public void setMensajeInformativoIconoLabel(JLabel mensajeInformativoIconoLabel) {
        this.mensajeInformativoIconoLabel = mensajeInformativoIconoLabel;
    }

    public JFrame getFramePrincipal() {
        return framePrincipal;
    }

    public void setFramePrincipal(JFrame framePrincipal) {
        this.framePrincipal = framePrincipal;
    }
}
```

B.6.15. hda.auth.gui.ValidarTrazoPanel.java

```
package hda.auth.gui;

import hda.auth.gui.listeners.ValidarTrazoListener;
import hda.auth.img.Trazo;

import java.awt.AWTException;
import java.awt.GraphicsConfiguration;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Robot;
import java.io.Serializable;
import java.util.ArrayList;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class ValidarTrazoPanel extends JPanel implements Serializable {

	private static final long serialVersionUID = 1L;
	private JDialog dialog;
	private int tipo_operacion;
	private Point inicioArrastre;
	private Point finArrastre;
	private ArrayList<Trazo> trazos = new ArrayList<Trazo>();
	private GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
	private GraphicsDevice[] gs = ge.getScreenDevices();

	/**
	 * Guarda una instancia del {@link JFrame} que lo contiene
	 */
	public ValidarTrazoPanel(JDialog dialog,int tipo_operacion){
		this.setDialog(dialog);
		this.setTipo_operacion(tipo_operacion);
		inicioArrastre = new Point(-1,-1);
		finArrastre = new Point(-1,-1);
		initComponents();
	}

	/**
	 * @return Devuelve un objeto de tipo {@link Point} que representa el inicio del arrastre del Mouse
	 */
	public Point getInicioArrastre() {
		return inicioArrastre;
	}

	public void setInicioArrastre(Point inicioArrastre) {
		this.inicioArrastre = inicioArrastre;
	}

	public Point getFinArrastre() {
		return finArrastre;
	}

	public void setFinArrastre(Point finArrastre) {
		this.finArrastre = finArrastre;
	}

	public ArrayList<Trazo> getTrazos() {
		return trazos;
	}
```

```
    public void setTrazo(ArrayList<Trazo> trazos) {
        this.trazos = trazos;
    }

    public void setDialog(JDialog dialog) {
        this.dialog = dialog;
    }

    public JDialog getDialog() {
        return dialog;
    }

    public Trazo getTrazoVacio(){
        Trazo trazo = new Trazo(getWidth(),getHeight());
        trazos.add(trazo);
        return trazo;
    }

    public void setTipo_operacion(int tipo_operacion) {
        this.tipo_operacion = tipo_operacion;
    }

    public int getTipo_operacion() {
        return tipo_operacion;
    }

    private void initComponents(){
        int xoffs = 0;
        int yoffs = 0;
        GraphicsDevice gd = null;

        gd = gs[0];
        GraphicsConfiguration[] gc = gd.getConfigurations();
        Rectangle gcBounds = gc[0].getBounds();
        xoffs = gcBounds.x;
        yoffs = gcBounds.y;

        setFocusable(true);
        setLocation(xoffs, yoffs);
        setSize(gc[0].getBounds().width, gc[0].getBounds().height);
        this.setLocation(0, 0);


        //Creamos y añadimos el listener de Eventos del ratón
        ValidarTrazoListener vtmel = new ValidarTrazoListener(this,getTipo_operacion());
        addMouseMotionListener(vtmel);//movimiento
        addMouseListener(vtmel);//botones

        try {
            Robot robot = new Robot();
            robot.mouseMove(getWidth()/2, getHeight()/2);
        } catch (AWTException e) {
            e.printStackTrace();
        }

    }

}
```

B.7. Paquete hda.auth.gui.listeners

B.7.1. hda.auth.gui.listeners.AdminPasswordSetupListener.java

```
package hda.auth.gui.listeners;

import hda.auth.cifrado.SecurityToolBox;
import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.dao.ExportarException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.gui.DialogAdminPasswordSetup;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.File;

public class AdminPasswordSetupListener implements WindowListener, ActionListener{
	DialogAdminPasswordSetup frmAdminPasswordSetup;
	public AdminPasswordSetupListener(DialogAdminPasswordSetup frmAdminPasswordSetup){
		this.frmAdminPasswordSetup = frmAdminPasswordSetup;
	}

	public void actionPerformed(ActionEvent evento) {
		//EVENTO VALIDAR TRAZO
		if(evento.getActionCommand().equals("Cambiar")){
			if(new String(frmAdminPasswordSetup.getClaveAccesoField().getPassword()).equals(new
String(frmAdminPasswordSetup.getClaveAccesoRepetirField().getPassword()))){
				try {
					//Calculamos SHA256 del password introducido
					String passwordHashStr = SecurityToolBox.getSHA256(new
String(frmAdminPasswordSetup.getClaveAccesoField().getPassword()));

					//Exportamos la clave de Admin
					ManejadorFicheros.exportar(passwordHashStr, new
File(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile()));

					//Mostramos mensaje en Pantalla
					frmAdminPasswordSetup.getMensajeInformativoLabel().setForeground(new
Color(0,200,50));
					frmAdminPasswordSetup.getMensajeInformativoLabel().setText("Clave de Administrador
cambiada");
				} catch (ConfigException e) {
					frmAdminPasswordSetup.getMensajeInformativoLabel().setText("Error de Configuración");
				} catch (ExportarException e) {
					frmAdminPasswordSetup.getMensajeInformativoLabel().setText("Error al cambiar la
clave");
				}
			}else{
				frmAdminPasswordSetup.getMensajeInformativoLabel().setText("Las claves deben ser iguales");
			}

		}

		//EVENTO REGISTRAR TRAZO
		if(evento.getActionCommand().equals("Cancelar")){
			frmAdminPasswordSetup.dispose();
		}

	}
```

```
	@Override
	public void windowActivated(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowClosed(WindowEvent arg0) {
		// TODO Auto-generated method stub
		frmAdminPasswordSetup.getFramePadre().setEnabled(true);
	}

	@Override
	public void windowClosing(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowDeactivated(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowDeiconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowIconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowOpened(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

}
```

B.7.2. hda.auth.gui.listeners.PasswordListener.java

```
package hda.auth.gui.listeners;

import hda.auth.cifrado.GeneradorClave;
import hda.auth.cifrado.SecurityToolBox;
import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.dao.ExportarException;
import hda.auth.dao.ImportarClaveException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.gui.DialogPassword;
import hda.auth.gui.DialogRegistrarTrazo;
import hda.auth.gui.PrincipalGUI;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.File;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class PasswordListener implements WindowListener, ActionListener{
	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	DialogPassword frmPassword;
	private int intentosPassword = 5;
	public PasswordListener(DialogPassword frmPassword){
		this.frmPassword = frmPassword;
	}

	public void actionPerformed(ActionEvent evento) {
		//EVENTO VALIDAR TRAZO
		if(evento.getActionCommand().equals("Entrar")){
			if(intentosPassword<=0){
				frmPassword.getMensajeInformativoLabel().setText("Acceso bloqueado. Contacte con el
Administrador");
				frmPassword.getMensajeInformativoLabel().setVisible(true);
			}else{
				if(frmPassword.getClaveAccesoField().getPassword()!=null && !("".equals(new
String(frmPassword.getClaveAccesoField().getPassword()))) ){
					try {
						//recuperamos la clave Admin
						String passStrStored = new String(ManejadorFicheros.importarClaveAdmin());

						//calculamos mediante SHA1 la huella del password introducido
					String passwordHashStr = SecurityToolBox.getSHA256(new
String(frmPassword.getClaveAccesoField().getPassword()));

					if(passwordHashStr.equals(passStrStored)){
						frmPassword.dispose();

							SwingUtilities.invokeLater(new Runnable() {
								public void run() {
									//creamos el JFrame donde dibujar el trazo a validar
									final JDialog dialogRegistrarTrazo = new
DialogRegistrarTrazo(PrincipalGUI.getInstance());

	dialogRegistrarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
					}
```

```
                                    });
                                    intentosPassword = 5;
                                }else{
                                    intentosPassword--;
                                    frmPassword.getMensajeInformativoLabel().setText("Password Incorrecta. Le quedan
"+intentosPassword+" intentos");
                                }
                                    if(intentosPassword==0){
                                        try {

    ManejadorFicheros.exportar(SecurityToolBox.getSHA256(GeneradorClave.generaClaveSecretaUID()),  new
File(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile()));
                                        } catch (ExportarException e) {
                                            frmPassword.getMensajeInformativoLabel().setText("Error de
Configuración");
                                            logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR
SEGURIDAD, CERRAMOS APLICACION");
                                            System.exit(1);
                                        } catch (ConfigException e) {
                                            frmPassword.getMensajeInformativoLabel().setText("Error de
Configuración");
                                            logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR
SEGURIDAD, CERRAMOS APLICACION");
                                            System.exit(1);
                                        }
                                        frmPassword.getMensajeInformativoLabel().setText("Acceso bloqueado. Póngase
en contacto con el Administrador");
                                        frmPassword.getMensajeInformativoLabel().setVisible(true);
                                    }
                                } catch (ImportarClaveException e) {
                                    try {

    ManejadorFicheros.exportar(SecurityToolBox.getSHA256(GeneradorClave.generaClaveSecretaUID()),  new
File(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile()));
                                } catch (ExportarException e1) {
                                    frmPassword.getMensajeInformativoLabel().setText("Error de Configuración");
                                    logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR SEGURIDAD,
CERRAMOS APLICACION");
                                    System.exit(1);
                                } catch (ConfigException e2) {
                                    frmPassword.getMensajeInformativoLabel().setText("Error de Configuración");
                                    logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR SEGURIDAD,
CERRAMOS APLICACION:"+e2.getMessage());
                                    System.exit(1);
                                }
                            }
                        }else{
                            frmPassword.getMensajeInformativoLabel().setText("Debe introducir una contraseña");
                        }
                    }

            }

            //EVENTO REGISTRAR TRAZO
            if(evento.getActionCommand().equals("Cancelar")){
                frmPassword.dispose();
            }

        }

        @Override
        public void windowActivated(WindowEvent arg0) {
            // TODO Auto-generated method stub

        }

        @Override
```

```
	public void windowClosed(WindowEvent arg0) {
		// TODO Auto-generated method stub
		frmPassword.getFramePadre().setEnabled(true);
	}

	@Override
	public void windowClosing(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowDeactivated(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowDeiconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowIconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void windowOpened(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

}
```

B.7.3. hda.auth.gui.listeners.PrincipalGUIListener.java

```
package hda.auth.gui.listeners;

import hda.auth.gui.PrincipalGUI;

import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class PrincipalGUIListener implements ChangeListener{

	PrincipalGUI principalGUI;
	public PrincipalGUIListener(PrincipalGUI principalGUI){
		super();
		this.principalGUI = principalGUI;
	}


	public void stateChanged(ChangeEvent evento){
		//limpiamos toda la plantilla
		if(principalGUI.getTabLoginPanel()!=null){
			principalGUI.getTabLoginPanel().limpiar();
		}
		if(principalGUI.getTabSetupPanel()!=null){
			principalGUI.getTabSetupPanel().limpiar();
		}


	}

}
```

B.7.4. hda.auth.gui.listeners.RegistrarTrazoListener.java

```
package hda.auth.gui.listeners;

import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.dao.ExportarLockException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.gui.PrincipalGUI;
import hda.auth.gui.RegistrarTrazoPanel;
import hda.auth.img.Trazo;
import hda.auth.locks.Lock;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.AWTException;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Robot;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.File;

import javax.swing.ImageIcon;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class RegistrarTrazoListener implements MouseMotionListener, MouseListener {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private RegistrarTrazoPanel panel;
	private Trazo trazoActivo;
	private String dispositivo;

	public RegistrarTrazoListener(RegistrarTrazoPanel panel){
		this.panel = panel;
		trazoActivo = panel.getTrazoVacio();
		try {
			dispositivo = LockFactory.getInstance().getDispositivo();
		} catch (LockFactoryConfigException e) {
			dispositivo = "udraw";
		}
	}

	@Override
	public void mouseDragged(MouseEvent evt) {
		//guardamos el evento en el Array que forma el trazo
		trazoActivo.addMouseEvent(evt);
		if("udraw".equals(dispositivo)){
			Graphics2D g2= (Graphics2D) trazoActivo.getImagen().getGraphics();
		g2.setStroke(new BasicStroke(5));
		g2.setColor(Color.WHITE);
		g2.drawLine(panel.getInicioArrastre().x, panel.getInicioArrastre().y,evt.getX(), evt.getY());
		panel.getGraphics().drawImage(trazoActivo.getImagen(),0,0,panel);
		g2.dispose();
		panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
		}
	}

	@Override
```

```
    public void mouseMoved(MouseEvent evt) {
        //guardamos el evento en el Array que forma el trazo
        trazoActivo.addMouseEvent(evt);
        if("chronosTI".equals(dispositivo)){
            if(panel.getInicioArrastre().x == -1 && panel.getInicioArrastre().y == -1 ){
                panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
            }
            Graphics2D g2= (Graphics2D) trazoActivo.getImagen().getGraphics();
        g2.setStroke(new BasicStroke(5));
        g2.setColor(Color.WHITE);
        g2.drawLine(panel.getInicioArrastre().x, panel.getInicioArrastre().y,evt.getX(), evt.getY());
        panel.getGraphics().drawImage(trazoActivo.getImagen(),0,0,panel);
        g2.dispose();
        panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
        }

    }

    @Override
    public void mouseClicked(MouseEvent arg0) {
        //guardamos el evento en el Array que forma el trazo
        trazoActivo.addMouseEvent(arg0);

        //El evento que define que el trazo ha finalizado en el botón central del ratón
        if(arg0.getButton()==2){
            //si necesitamos mas trazos
            int numero_trazos = 1;
            try {
                numero_trazos = LockFactory.getInstance().getNumero_trazos();
            } catch (LockFactoryConfigException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            logger.info("Trazo "+panel.getTrazos().size()+" de "+numero_trazos);
            if(panel.getTrazos().size()<numero_trazos){
                try {
                    Robot robot = new Robot();
                    robot.mouseMove(panel.getWidth()/2, panel.getHeight()/2);
                    panel.setInicioArrastre(new Point(-1,-1));
                } catch (AWTException e) {
                    e.printStackTrace();
                }
                //pedimos trazo nuevo vacio
                trazoActivo = panel.getTrazoVacio();
                //limpiamos el panel para volver a pintar
                panel.setVisible(false);
                panel.setVisible(true);

            }else{
                panel.setVisible(false);
                panel.getDialog().dispose();
            try {

                Lock lock = LockFactory.getInstance().createLock(panel.getTrazos());
                PrincipalGUI.getInstance().setLock(lock);

                //exportamos a fichero el lock
                File lockFile = new
File(Config.getInstance().getModuleConfigByName("LockObject").getConfigFile());
                ManejadorFicheros.exportar(lock, lockFile);

                PrincipalGUI.getInstance().getTabLoginPanel().getValidarTrazoBoton().setEnabled(true);
                PrincipalGUI.getInstance().getTabLoginPanel().getRegistrarTrazoResLabel().setIcon(new
ImageIcon("images"+File.separator+"ok.png"));

    PrincipalGUI.getInstance().getTabLoginPanel().getRegistrarTrazoResLabel().setVisible(true);
```

```
                        /*
                        //Obtenemos los bytes que forman la imagen
                        ByteArrayOutputStream baos = new ByteArrayOutputStream();
                        ImageIO.write(panel.getTrazo().getImagen(), "bmp", baos);
                        byte[] imagenBytes = baos.toByteArray();
                        */
                        } catch (LockFactoryConfigException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        } catch (ConfigException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        } catch (ExportarLockException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  }


            }
      }

      @Override
      public void mouseEntered(MouseEvent e) {

      }

      @Override
      public void mouseExited(MouseEvent e) {

      }

      @Override
      public void mousePressed(MouseEvent e) {
            panel.setInicioArrastre(new Point(e.getX(), e.getY()));
         //panel.repaint();
      }

      @Override
      public void mouseReleased(MouseEvent e) {

      }
      /*
   private BufferedImage getScreenShot(JPanel panel){
      int w = panel.getWidth();
      int h = panel.getHeight();
      BufferedImage bi = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
      Graphics2D g = bi.createGraphics();
      panel.print(g);
      return bi;
   }
   */

}
```

B.7.5. hda.auth.gui.listeners.TabLoginListener.java

```
package hda.auth.gui.listeners;

import hda.auth.config.Config;
import hda.auth.dao.ImportarClaveException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.gui.DialogMensajeTemporal;
import hda.auth.gui.DialogPassword;
import hda.auth.gui.DialogValidarTrazo;
import hda.auth.gui.PrincipalGUI;
import hda.auth.gui.TabLoginPanel;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.File;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class TabLoginListener implements ActionListener, MouseListener{
	TabLoginPanel tabLoginPanel;
	public TabLoginListener(TabLoginPanel tabLoginPanel){
		this.tabLoginPanel = tabLoginPanel;
	}

	public void actionPerformed(ActionEvent evento) {
		tabLoginPanel.getValidarTrazoResLabel().setVisible(false);
		//EVENTO VALIDAR TRAZO
		if(evento.getActionCommand().equals("validarTrazo")){
			//limpiamos toda la plantilla
			if(tabLoginPanel!=null){
				tabLoginPanel.limpiar();
			}
			try{
				ManejadorFicheros.importarClavePrivada();
				SwingUtilities.invokeLater(new Runnable() {
					public void run() {
						//creamos el JFrame donde dibujar el trazo a validar
						final JDialog dialogValidarTrazo = new
DialogValidarTrazo(PrincipalGUI.getInstance(),DialogValidarTrazo.TipoOperacion.LOGIN);
						dialogValidarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
					}
				});
			}catch(ImportarClaveException ice){
				new DialogMensajeTemporal(PrincipalGUI.getInstance(),3, "Debe Generar los
Certificados",DialogMensajeTemporal.TipoIcono.FLECHA);
			}

		}

		//EVENTO REGISTRAR TRAZO
		if(evento.getActionCommand().equals("registrarTrazo")){
			//si existe trazo previo comprobamos antes de cambiarlo
			if(PrincipalGUI.getInstance().getLock()!=null){
				new DialogMensajeTemporal(PrincipalGUI.getInstance(),3, "Introduzca Trazo
Anterior",DialogMensajeTemporal.TipoIcono.FLECHA);
				SwingUtilities.invokeLater(new Runnable() {
					public void run() {
						//creamos el JFrame donde dibujar el trazo a validar
						final JDialog dialogValidarTrazo = new
DialogValidarTrazo(PrincipalGUI.getInstance(),DialogValidarTrazo.TipoOperacion.CAMBIO_TRAZO);
```

```
                                dialogValidarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                            }
                        });
                    }else{
                        SwingUtilities.invokeLater(new Runnable() {
                            public void run() {
                                //pedimos el password para primer registro de trazo
                                final DialogPassword frmValidarTrazo = new
DialogPassword(PrincipalGUI.getInstance());
                                frmValidarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                            }
                        });
                    }
                }
                //EVENTO LOGOUT
                if(evento.getActionCommand().equals("logout")){
                    //eliminamos token de login
                    try {
                        new DialogMensajeTemporal(PrincipalGUI.getInstance(),3, "Realizando
Logout...",DialogMensajeTemporal.TipoIcono.WAIT);
                        File tokenFile = new
File(Config.getInstance().getModuleConfigByName("LoginObject").getConfigFile());
                        tokenFile.delete();
                        new DialogMensajeTemporal(PrincipalGUI.getInstance(),3, "Logout
Realizado",DialogMensajeTemporal.TipoIcono.PAPELERA);
                    } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                    //GeneradorClave.generaClavePrivada();
                    //cerramos la ventana de aplicacion
                    //PrincipalGUI.getInstance().dispose();
                }
            }

            @Override
            public void mouseClicked(MouseEvent arg0) {
                //EVENTO VALIDAR TRAZO
                if(arg0.getButton()==2){
                    tabLoginPanel.getValidarTrazoResLabel().setVisible(false);
                    //limpiamos toda la plantilla
                    if(tabLoginPanel!=null){
                        tabLoginPanel.limpiar();
                    }

                    SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                            //creamos el JFrame donde dibujar el trazo a validar
                            final JDialog dialogValidarTrazo = new
DialogValidarTrazo(PrincipalGUI.getInstance(),DialogValidarTrazo.TipoOperacion.LOGIN);
                            dialogValidarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        }
                    });
                }


            }

            @Override
            public void mouseEntered(MouseEvent arg0) {
                // TODO Auto-generated method stub

            }

            @Override
            public void mouseExited(MouseEvent arg0) {
                // TODO Auto-generated method stub
```

```
	}

	@Override
	public void mousePressed(MouseEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void mouseReleased(MouseEvent arg0) {
		// TODO Auto-generated method stub

	}

}
```

B.7.6. hda.auth.gui.listeners.TabSetupListener.java

```
package hda.auth.gui.listeners;

import hda.auth.cifrado.GeneradorClave;
import hda.auth.cifrado.SecurityToolBox;
import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.dao.ExportarClaveException;
import hda.auth.dao.ExportarException;
import hda.auth.dao.GenerarClaveException;
import hda.auth.dao.ImportarClaveException;
import hda.auth.dao.ManejadorFicheros;
import hda.auth.gui.DialogAdminPasswordSetup;
import hda.auth.gui.DialogMensajeTemporal;
import hda.auth.gui.PrincipalGUI;
import hda.auth.gui.TabSetupPanel;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import javax.swing.filechooser.FileNameExtensionFilter;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class TabSetupListener implements ActionListener {
	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	private int intentosPassword = 5;
	TabSetupPanel tabSetupPanel;
	public TabSetupListener(TabSetupPanel tabSetupPanel){
		this.tabSetupPanel = tabSetupPanel;
	}

	@Override
	public void actionPerformed(ActionEvent evento) {
		//EVENTO GENERAR PAREJA CLAVES
		if(evento.getActionCommand().equals("GenerarCertificados")){
			//limpiamos toda la plantilla
			if(tabSetupPanel!=null){
				tabSetupPanel.limpiar();
			}
			String directorioClaves = "data";

			JFileChooser explorador = new JFileChooser(directorioClaves+File.separator);
			explorador.setDialogTitle("Exportar Clave Publica...");
			//explorador.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
			String[] extensiones = {"key"};
			explorador.setFileFilter(new FileNameExtensionFilter("HDA Key",extensiones));

			int seleccion = explorador.showSaveDialog(tabSetupPanel);
			switch(seleccion) {
			case JFileChooser.APPROVE_OPTION:

				try {
					//Llamada a generar que en funcion de si el objeto es de tipo TipoCifradoSimetrico o
TipoCifradoAsimetrico
```

```
    //ManejadorFicheros.exportar(claseCifradoImplement.generar(),explorador.getSelectedFile());


    ManejadorFicheros.exportar(SecurityToolBox.generarParejaClavesAleatoria(),explorador.getSelectedFile());
                        tabSetupPanel.getMensajeInformativoLabel().setFont(new java.awt.Font("Arial
Black",0,14));
                        tabSetupPanel.getMensajeInformativoLabel().setForeground(new Color(0,200,50));
                        tabSetupPanel.getMensajeInformativoLabel().setText("Claves exportadas correctamente");
                        tabSetupPanel.getMensajeInformativoLabel().setVisible(true);
                        tabSetupPanel.getMensajeInformativoIconoLabel().setIcon(new
ImageIcon("images"+File.separator+"ok.png"));
                        tabSetupPanel.getMensajeInformativoIconoLabel().setVisible(true);
                    } catch (ExportarClaveException e) {
                        tabSetupPanel.getMensajeInformativoLabel().setFont(new java.awt.Font("Arial
Black",0,14));
                        tabSetupPanel.getMensajeInformativoLabel().setForeground(new Color(200,0,50));
                        tabSetupPanel.getMensajeInformativoLabel().setText("Error al exportar claves");
                        tabSetupPanel.getMensajeInformativoLabel().setVisible(true);
                        tabSetupPanel.getMensajeInformativoIconoLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));
                        tabSetupPanel.getMensajeInformativoIconoLabel().setVisible(true);
                    }catch (GenerarClaveException gce){
                        tabSetupPanel.getMensajeInformativoLabel().setFont(new java.awt.Font("Arial
Black",0,14));
                        tabSetupPanel.getMensajeInformativoLabel().setForeground(new Color(200,0,50));
                        tabSetupPanel.getMensajeInformativoLabel().setText("Error al exportar claves");
                        tabSetupPanel.getMensajeInformativoLabel().setVisible(true);
                        tabSetupPanel.getMensajeInformativoIconoLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));
                        tabSetupPanel.getMensajeInformativoIconoLabel().setVisible(true);
                    }
                    break;

                case JFileChooser.CANCEL_OPTION:
                    tabSetupPanel.getMensajeInformativoLabel().setFont(new java.awt.Font("Arial Black",0,14));
                    tabSetupPanel.getMensajeInformativoLabel().setForeground(new Color(0,64,128));
                    tabSetupPanel.getMensajeInformativoLabel().setText("Operacion Cancelada");
                    tabSetupPanel.getMensajeInformativoLabel().setVisible(true);
                    tabSetupPanel.getMensajeInformativoIconoLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));
                    tabSetupPanel.getMensajeInformativoIconoLabel().setVisible(true);
                    break;

                case JFileChooser.ERROR_OPTION:
                    tabSetupPanel.getMensajeInformativoLabel().setFont(new java.awt.Font("Arial Black",0,14));
                    tabSetupPanel.getMensajeInformativoLabel().setForeground(new Color(200,0,50));
                    tabSetupPanel.getMensajeInformativoLabel().setText("Error al exportar claves");
                    tabSetupPanel.getMensajeInformativoLabel().setVisible(true);
                    tabSetupPanel.getMensajeInformativoIconoLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));
                    tabSetupPanel.getMensajeInformativoIconoLabel().setVisible(true);
                    break;
                }

            }

            //EVENTO VALIDAR ADMIN
            if(evento.getActionCommand().equals("Entrar")){
                try {
                    if(intentosPassword <=0){
                        tabSetupPanel.getClaveAccesoResLabel().setText("Acceso bloqueado.");
                        tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                    }else{
                        //recuperamos la clave Admin
                        String passStrStored = new String(ManejadorFicheros.importarClaveAdmin());
                        //calculamos mediante SHA256 la huella del password introducido
```

```
                        String passwordHashStr = SecurityToolBox.getSHA256(new
String(tabSetupPanel.getClaveAccesoField().getPassword()));

                    if(passwordHashStr.equals(passStrStored)){
                        tabSetupPanel.setAdminLogged(true);
                        tabSetupPanel.initGUI();
                        intentosPassword = 5;
                    }else{
                      intentosPassword--;
                        tabSetupPanel.setAdminLogged(false);
                        //tabSetupPanel.initGUI();
                        tabSetupPanel.getClaveAccesoResLabel().setText("Password Incorrecta. Le quedan
"+intentosPassword+" intentos");
                        tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                        final int tiempoPenalizacion = (5-intentosPassword)*5;
                        new DialogMensajeTemporal(PrincipalGUI.getInstance(),tiempoPenalizacion,
"Password Incorrecta",DialogMensajeTemporal.TipoIcono.FLECHA).setDefaultCloseOperation(0);
                    }
                      if(intentosPassword==0){
                        try {

      ManejadorFicheros.exportar(SecurityToolBox.getSHA256(GeneradorClave.generaClaveSecretaUID()),  new
File(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile()));
                            new DialogMensajeTemporal(PrincipalGUI.getInstance(),3, "Clave
Reestablecida",DialogMensajeTemporal.TipoIcono.FLECHA);
                        } catch (ExportarException e) {
                            tabSetupPanel.getClaveAccesoResLabel().setText("Error de Configuración");
                            logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR SEGURIDAD,
CERRAMOS APLICACION");
                            System.exit(1);
                        }
                        tabSetupPanel.getClaveAccesoResLabel().setText("Acceso bloqueado.");
                        tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                      }
                    }


                } catch (ConfigException e) {
                    tabSetupPanel.setAdminLogged(false);
                    //tabSetupPanel.initGUI();
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error de Configuración");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.setAdminLogged(false);
                    //tabSetupPanel.initGUI();
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error formato clave");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                } catch (ImportarClaveException e) {
                    try {

      ManejadorFicheros.exportar(SecurityToolBox.getSHA256(GeneradorClave.generaClaveSecretaUID()),  new
File(Config.getInstance().getModuleConfigByName("AdminPasswordHASH").getConfigFile()));
                        tabSetupPanel.getClaveAccesoResLabel().setText("Clave Inicial Reestablecida");
                        tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                    } catch (ExportarException e2) {
                        tabSetupPanel.getClaveAccesoResLabel().setText("Error de Configuración");
                        logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR SEGURIDAD, CERRAMOS
APLICACION:"+e2.getMessage());
                        System.exit(1);
                    } catch (ConfigException e3) {
                        tabSetupPanel.getClaveAccesoResLabel().setText("Error de Configuración");
                        logger.fatal("ERROR GENERANDO CLAVE ADMIN... POR SEGURIDAD, CERRAMOS
APLICACION:"+e3.getMessage());
                        System.exit(1);
                    }
                }
```

```
        }
        //EVENTO FINALIZAR CONFIGURACION ADMIN
        if(evento.getActionCommand().equals("GenerarClaveAdmin")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                try {
                    SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                            //creamos el JFrame donde dibujar el trazo a validar
                            DialogAdminPasswordSetup frmAdminPasswordSetup = new
DialogAdminPasswordSetup(tabSetupPanel.getFramePrincipal());

    frmAdminPasswordSetup.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        }
                    });

                } catch (Exception e) {
                    logger.error("Error al procesar Evento GenerarClaveAdmin:"+e.getMessage());
                }
            }
        }


        //EVENTO GUARDAR CONFIGURACION ADMIN
        if(evento.getActionCommand().equals("Guardar")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                try {
                    LockFactory.getInstance().saveConfig();
                } catch (LockFactoryConfigException e) {
                    logger.error("Error al procesar Evento Guardar:"+e.getMessage());
                }
            }
        }
        //EVENTO FINALIZAR CONFIGURACION ADMIN
        if(evento.getActionCommand().equals("SalirConfig")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                try {
                    tabSetupPanel.setAdminLogged(false);
                    tabSetupPanel.initGUI();
                } catch (LockFactoryConfigException e) {
                    logger.error("Error al procesar Evento SalirConfig:"+e.getMessage());
                }
            }
        }
        //EVENTO SELECCION DISPOSITIVO UDRAW
        if(evento.getActionCommand().equals("udrawDevice")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setDispositivo("udraw");
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando dispositivo");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION DISPOSITIVO CHRONOS
        if(evento.getActionCommand().equals("chronosDevice")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setDispositivo("chronosTI");
                } catch (LockFactoryConfigException e) {
```

```
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando dispositivo");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION VALIDEZ 1
        if(evento.getActionCommand().equals("horasvalidez1")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setHoras_validez(1);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando horas validez");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION VALIDEZ 2
        if(evento.getActionCommand().equals("horasvalidez2")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setHoras_validez(6);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando horas validez");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION VALIDEZ 3
        if(evento.getActionCommand().equals("horasvalidez3")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setHoras_validez(12);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando horas validez");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION VALIDEZ 4
        if(evento.getActionCommand().equals("horasvalidez4")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setHoras_validez(24);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando horas validez");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
                }
            }
        }
        //EVENTO SELECCION VALIDEZ 5
        if(evento.getActionCommand().equals("horasvalidez5")){
            //limpiamos toda la plantilla
            if(tabSetupPanel!=null){
                tabSetupPanel.limpiar();
                try {
                    LockFactory.getInstance().setHoras_validez(48);
                } catch (LockFactoryConfigException e) {
                    tabSetupPanel.getClaveAccesoResLabel().setText("Error Seleccionando horas validez");
                    tabSetupPanel.getClaveAccesoResLabel().setVisible(true);
```

```
                        }
                    }
                }
                //EVENTO ---
                if(evento.getActionCommand().equals("---")){
                    //limpiamos toda la plantilla
                    if(tabSetupPanel!=null){
                        tabSetupPanel.limpiar();
                    }
                }
            }
        }
```

B.7.7. hda.auth.gui.listeners.ValidarTrazoListener.java

```
package hda.auth.gui.listeners;

import hda.auth.config.parsers.LoginGenerationException;
import hda.auth.config.parsers.XMLloginParser;
import hda.auth.gui.DialogMensajeTemporal;
import hda.auth.gui.DialogRegistrarTrazo;
import hda.auth.gui.DialogValidarTrazo;
import hda.auth.gui.PrincipalGUI;
import hda.auth.gui.ValidarTrazoPanel;
import hda.auth.img.Trazo;
import hda.auth.keys.Key;
import hda.auth.keys.KeyFactory;
import hda.auth.keys.KeyFactoryConfigException;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import java.awt.AWTException;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Robot;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.File;

import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class ValidarTrazoListener implements MouseMotionListener, MouseListener {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());
	private ValidarTrazoPanel panel;
	private int tipo_operacion;
	private Trazo trazoActivo;
	private String dispositivo;

	public ValidarTrazoListener(ValidarTrazoPanel panel,int tipo_operacion){
		this.panel = panel;
		this.tipo_operacion = tipo_operacion;
		trazoActivo = panel.getTrazoVacio();
		try {
			dispositivo = LockFactory.getInstance().getDispositivo();
		} catch (LockFactoryConfigException e) {
			dispositivo = "udraw";
		}
	}

	@Override
	public void mouseDragged(MouseEvent evt) {
		//guardamos el evento en el Array que forma el trazo
		trazoActivo.addMouseEvent(evt);
		if("udraw".equals(dispositivo)){
			Graphics2D g2= (Graphics2D) trazoActivo.getImagen().getGraphics();
		g2.setStroke(new BasicStroke(5));
		g2.setColor(Color.WHITE);
		g2.drawLine(panel.getInicioArrastre().x, panel.getInicioArrastre().y,evt.getX(), evt.getY());
```

```
            panel.getGraphics().drawImage(trazoActivo.getImagen(),0,0,panel);
            g2.dispose();
            panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
            }
        }

        @Override
        public void mouseMoved(MouseEvent evt) {
            //guardamos el evento en el Array que forma el trazo
            trazoActivo.addMouseEvent(evt);
            if("chronosTI".equals(dispositivo)){
                if(panel.getInicioArrastre().x == -1 && panel.getInicioArrastre().y == -1 ){
                    panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
                }
                Graphics2D g2= (Graphics2D) trazoActivo.getImagen().getGraphics();
            g2.setStroke(new BasicStroke(5));
            g2.setColor(Color.WHITE);
            g2.drawLine(panel.getInicioArrastre().x, panel.getInicioArrastre().y,evt.getX(), evt.getY());
            panel.getGraphics().drawImage(trazoActivo.getImagen(),0,0,panel);
            g2.dispose();
            panel.setInicioArrastre(new Point(evt.getX(),evt.getY()));
            }

        }

        @Override
        public void mouseClicked(MouseEvent arg0) {
            //guardamos el evento en el Array que forma el trazo
            trazoActivo.addMouseEvent(arg0);
            if(arg0.getButton()==2){
                //si necesitamos mas trazos
                int numero_trazos = 1;
                try {
                    numero_trazos = KeyFactory.getInstance().getNumero_trazos();
                } catch (KeyFactoryConfigException e) {
                    e.printStackTrace();
                }
                logger.info("Trazo "+panel.getTrazos().size()+" de "+numero_trazos);
                if(panel.getTrazos().size()<numero_trazos){
                    try {
                        Robot robot = new Robot();
                        robot.mouseMove(panel.getWidth()/2, panel.getHeight()/2);
                        panel.setInicioArrastre(new Point(-1,-1));
                    } catch (AWTException e) {
                        e.printStackTrace();
                    }
                    //pedimos trazo nuevo vacio
                    trazoActivo = panel.getTrazoVacio();
                    //limpiamos el panel para volver a pintar
                    panel.setVisible(false);
                    panel.setVisible(true);


                }else{
                    panel.setVisible(false);
                    panel.getDialog().dispose();
                try {

                    Key key = KeyFactory.getInstance().createKey(panel.getTrazos());
                    PrincipalGUI.getInstance().setKey(key);

                    boolean abierto =
PrincipalGUI.getInstance().getLock().open(PrincipalGUI.getInstance().getKey());

                    //Si la validacion es correcta
                    if(abierto){
                        switch(tipo_operacion){
```

```
                        case DialogValidarTrazo.TipoOperacion.LOGIN:
                            //JOptionPane.showMessageDialog(panel, "EL CANDADO SE HA ABIERTO", null,
JOptionPane.INFORMATION_MESSAGE);
                            logger.info("EL CANDADO SE HA ABIERTO!!!! :) ");

    PrincipalGUI.getInstance().getTabLoginPanel().getValidarTrazoResLabel().setIcon(new
ImageIcon("images"+File.separator+"ok.png"));

    PrincipalGUI.getInstance().getTabLoginPanel().getValidarTrazoResLabel().setVisible(true);
                            //Generamos XML con login OK
                            XMLloginParser xmlloginParser = new XMLloginParser();
                            xmlloginParser.generaLogin();
                            break;
                        case DialogValidarTrazo.TipoOperacion.CAMBIO_TRAZO:
                            SwingUtilities.invokeLater(new Runnable() {
                                public void run() {
                                    new DialogMensajeTemporal(PrincipalGUI.getInstance(), 3, "Ahora...
Registre su Trazo",DialogMensajeTemporal.TipoIcono.FLECHA );
                                    //creamos el JFrame donde dibujar el trazo a validar
                                    final JDialog dialogRegistrarTrazo = new
DialogRegistrarTrazo(PrincipalGUI.getInstance());

    dialogRegistrarTrazo.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                                }
                            });
                            break;
                        }

                    }else{
                        switch(tipo_operacion){
                        case DialogValidarTrazo.TipoOperacion.LOGIN:
                            //JOptionPane.showMessageDialog(panel, "EL CANDADO PERMANECE
CERRADO", null, JOptionPane.ERROR_MESSAGE);
                            logger.info("EL CANDADO PERMANECE CERRADO :( ");

    PrincipalGUI.getInstance().getTabLoginPanel().getValidarTrazoResLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));

    PrincipalGUI.getInstance().getTabLoginPanel().getValidarTrazoResLabel().setVisible(true);
                                //Generamos XML con login NOK
                            break;
                        case DialogValidarTrazo.TipoOperacion.CAMBIO_TRAZO:

    PrincipalGUI.getInstance().getTabLoginPanel().getRegistrarTrazoResLabel().setIcon(new
ImageIcon("images"+File.separator+"nok.png"));

    PrincipalGUI.getInstance().getTabLoginPanel().getRegistrarTrazoResLabel().setVisible(true);
                            break;
                        }

                    }

                    /*
                    //Obtenemos los bytes que forman la imagen
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();
                    ImageIO.write(panel.getTrazo().getImagen(), "bmp", baos);
                    byte[] imagenBytes = baos.toByteArray();


                    } catch (IOException ioe) {
                        ioe.printStackTrace(); */
                    } catch (KeyFactoryConfigException e) {
                        e.printStackTrace();
                    } catch (LoginGenerationException e) {
                        e.printStackTrace();
                    }
                }
```

```
            }
        }

        @Override
        public void mouseEntered(MouseEvent e) {

        }

        @Override
        public void mouseExited(MouseEvent e) {

        }

        @Override
        public void mousePressed(MouseEvent e) {
              panel.setInicioArrastre(new Point(e.getX(), e.getY()));
           //panel.repaint();
        }

        @Override
        public void mouseReleased(MouseEvent e) {

        }

}
```

B.8. Paquete hda.auth.img

B.8.1. hda.auth.img.SelectorPantalla.java

```
package hda.auth.img;

import java.awt.GraphicsConfiguration;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;
import java.util.ArrayList;

import javax.swing.JFrame;


/**
 * Clase que gestiona los múltiples monitores que se pueden conectar a un mismo equipo.
 * Contiene las funciones básicas para crear objetos de tipo {@link JFrame} no visibles vinculados con los distintos monitores.
*/
public class SelectorPantalla {
	private static SelectorPantalla selectorPantalla;
	private GraphicsEnvironment ge;
	private GraphicsDevice[] gs;


	/**
	 * Método que devuelve una instancia de la clase {@link SelectorPantalla}.</br>
	 * Este método está implementado como un singleton, devuelve una instancia de la clase {@link SelectorPantalla}.
	 *
	 * @return devuelve un {@link SelectorPantalla}.
	 */
	public static SelectorPantalla getInstance(){
		if(selectorPantalla==null){
			selectorPantalla=new SelectorPantalla();
		}
		return selectorPantalla;
	}
	/**
	 * Constructor de la clase.
	 *
	 */
	private SelectorPantalla(){
		ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
		gs = ge.getScreenDevices();
	}

	/**
	 * Este método devuelve el número de pantallas conectadas al equipo.</br></br>
	 *
	 *  Un ejemplo de uso es el siguiente:</br></br>
	 *  <code>
	 *  <pre>
	 *  SelectorPantalla selectorPantalla = SelectorPantalla.getInstance();
	 *  System.out.println("Numero de pantallas = "+selectorPantalla.getNumeroPantallas());
	 *  </pre>
	 *  </code>
	 *
	 * @return devuelve un int con el numero de pantallas conectadas al equipo.
	 */
	public int getNumeroPantallas(){
		return gs.length;
	}
```

```
    /**
     * Crea un {@link JFrame} en una pantalla pasada como parámetro.</br></br>
     *
     * Un ejemplo de uso para una sola pantalla es el siguiente:</br></br>
     * <code>
     * <pre>
     *  SelectorPantalla selectorPantalla = SelectorPantalla.getInstance();
     *  JFrame f = null;
     *  try {
     *     f = selectorPantalla.crearFrameEnPantalla(1);
     *     JLabel jlabel = new JLabel("Esta es la PANTALLA=1");
     *     f.getContentPane().add(jlabel);
     *     f.pack();
     *     f.show();
     *  } catch (SelectorPantallaException e) {
     *     e.printStackTrace();
     *  }
     * </pre>
     * </code>
     *
     * @param numeroPantalla numeradas de 1..n.
     * @return devuelve un {@link JFrame} no visible situado en la esquina superior izquierda de la pantalla pasada
como parámetro.
     * @exception SelectorPantallaException en caso de provocar algún error.
     */
    public JFrame crearFrameEnPantalla(int numeroPantalla) throws SelectorPantallaException{
        GraphicsDevice gd = null;
        try{
            gd = gs[numeroPantalla-1];
            GraphicsConfiguration[] gc = gd.getConfigurations();
            Rectangle gcBounds = gc[0].getBounds();
            int xoffs = gcBounds.x;
            int yoffs = gcBounds.y;
            JFrame f = new JFrame(gs[numeroPantalla-1].getDefaultConfiguration());
            f.setLocation(xoffs,yoffs);
            return f;
        }catch(IndexOutOfBoundsException iob){
            throw new SelectorPantallaException("No existe la pantalla numero "+numeroPantalla);
        }catch(Exception e){
            throw new SelectorPantallaException("Error genérico creando frame en pantalla "+numeroPantalla);
        }

    }

    public JFrame crearFrameEnPantalla(int numeroPantalla,int caso) throws SelectorPantallaException{
        GraphicsDevice gd = null;
        System.out.print("crearFrameEnPantalla");
        try{
            JFrame f=null;
            System.out.print("crear JFrame");
            gd = gs[numeroPantalla-1];
            GraphicsConfiguration[] gc = gd.getConfigurations();
            System.out.print("Obtiene configuración");
            Rectangle gcBounds = gc[0].getBounds();
            int xoffs = gcBounds.x;
            int yoffs = gcBounds.y;
            if (caso== 1){
                System.out.print("Caso 1");
             f = new JFrame(gs[numeroPantalla-1].getDefaultConfiguration());
            }else if (caso== 2){
                System.out.print("Caso 2");
             f = new JFrame(gs[numeroPantalla-1].getDefaultConfiguration());
             }
            f.setLocation(xoffs,yoffs);
            return f;
        }catch(IndexOutOfBoundsException iob){
```

```
                System.out.print("Error creando pantalla");
                throw new SelectorPantallaException("No existe la pantalla numero "+numeroPantalla);

            }catch(Exception e){
                System.out.print("Error genérico creando frame pantalla");
                throw new SelectorPantallaException("Error genérico creando frame en pantalla "+numeroPantalla);
            }

        }
        /*public JFrmMain2 crearFrameEnPantalla2(int numeroPantalla,int caso) throws SelectorPantallaException{
            GraphicsDevice gd = null;
            try{
                gd = gs[numeroPantalla-1];
                GraphicsConfiguration[] gc = gd.getConfigurations();
                Rectangle gcBounds = gc[0].getBounds();
                int xoffs = gcBounds.x;
                int yoffs = gcBounds.y;
                JFrmMain2 f = new JFrmMain2(gs[numeroPantalla-1].getDefaultConfiguration());
                f.setLocation(xoffs,yoffs);
                return f;
            }catch(IndexOutOfBoundsException iob){
                throw new SelectorPantallaException("No existe la pantalla numero "+numeroPantalla);
            }catch(Exception e){
                throw new SelectorPantallaException("Error genérico creando frame en pantalla "+numeroPantalla);
            }

        }*/
        /**
         * Crea un {@link JFrame} en todas las pantallas del sistema.</br></br>
         *
         * Un ejemplo de uso para varias pantallas es el siguiente:</br></br>
         * <code>
         * <pre>
         * SelectorPantalla selectorPantalla = SelectorPantalla.getInstance();
         * ArrayList listaFrames = null;
         * try {
         *     listaFrames = selectorPantalla.crearFramesEnPantallas();
         *     for(int i = 0;i < selectorPantalla.getNumeroPantallas();i++){
         *         JLabel jlabel = new JLabel("Esta es la PANTALLA="+(i+1));
         *         ((JFrame)listaFrames.get(i)).getContentPane().add(jlabel);
         *         ((JFrame)listaFrames.get(i)).pack();
         *         ((JFrame)listaFrames.get(i)).show();
         *     }
         * } catch (SelectorPantallaException e) {
         *     e.printStackTrace();
         * }
         * </pre>
         * </code>
         *
         * @return devuelve un {@link ArrayList} de {@link JFrame} no visibles cada frame se sitúa en una pantalla
distinta en la esquina superior izquierda
         * @exception SelectorPantallaException en caso de provocar algún error.
         */
        public ArrayList<JFrame> crearFramesEnPantallas() throws SelectorPantallaException{
            ArrayList<JFrame>  listaPantallas = new ArrayList<JFrame> ();
            int j=0;
            try{
                for(j=0;j<gs.length;j++){
                    GraphicsDevice gd = gs[j];
                    GraphicsConfiguration[] gc = gd.getConfigurations();
                    Rectangle gcBounds = gc[0].getBounds();
                    int xoffs = gcBounds.x;
                    int yoffs = gcBounds.y;
                    JFrame f = new JFrame(gs[j].getDefaultConfiguration());
                    f.setLocation(xoffs,yoffs);
                    listaPantallas.add(f);
                }
```

```
                return listaPantallas;
            }catch(Exception e){
                throw new SelectorPantallaException("Error genérico creando frame en pantalla "+(j+1));
            }
        }
}
```

B.8.2. hda.auth.img.SelectorPantallaException.java

```
package hda.auth.img;

/**
 * Clase que representa las Excepciones lanzadas por {@link SelectorPantalla}.
*/
public class SelectorPantallaException extends Exception {

    private static final long serialVersionUID = 1L;

    /**
     * Constructor de la clase.
     *
     */
    public SelectorPantallaException(String msg) {
        super(msg);
    }
}
```

B.8.3. hda.auth.img.Trazo.java

```
package hda.auth.img;

import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.util.ArrayList;

public class Trazo {

    private BufferedImage imagen;
    private ArrayList<MouseEvent> mouseEvents = new ArrayList<MouseEvent>();

    public Trazo(int width,int height){
        imagen = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    }

    public BufferedImage getImagen() {
        return imagen;
    }

    public void setImagen(BufferedImage imagen) {
        this.imagen = imagen;
    }

    public synchronized void addMouseEvent(MouseEvent mouseEvent){
        mouseEvents.add(mouseEvent);
    }
    public synchronized ArrayList<MouseEvent> getMouseEvents(){
        return mouseEvents;
    }

}
```

B.9. Paquete hda.auth.keys

B.9.1. hda.auth.keys.Key.java

```
package hda.auth.keys;

import java.util.ArrayList;

import hda.auth.parameters.iKeyParameter;

public class Key {
	private ArrayList<iKeyParameter> parametros;

	public ArrayList<iKeyParameter> getParametros() {
		return parametros;
	}

	public void setParametros(ArrayList<iKeyParameter> parametros) {
		this.parametros = parametros;
	}
}
```

B.9.2. hda.auth.keys.KeyFactory.java

```
package hda.auth.keys;

import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.config.bean.KeyFactoryParameterConfigBean;
import hda.auth.config.parsers.XMLkeyFactoryParser;
import hda.auth.img.Trazo;
import hda.auth.keys.KeyFactory;
import hda.auth.keys.KeyFactoryConfigException;
import hda.auth.parameters.iKeyParameter;

import java.io.File;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class KeyFactory {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private static KeyFactory inst;
	private int numero_trazos = 1;
	private ArrayList<KeyFactoryParameterConfigBean> key_params_configurations = new
ArrayList<KeyFactoryParameterConfigBean>();


	public static KeyFactory getInstance() throws KeyFactoryConfigException{
		if(inst==null){
			inst = new KeyFactory();
		}
		return inst;
	}

	public KeyFactory() throws KeyFactoryConfigException {
		File configuracion;
		try {
			configuracion = new
File(Config.getInstance().getModuleConfigByName("KeyFactory").getConfigFile());
			if(!configuracion.exists()){
				throw new KeyFactoryConfigException("Fichero:
"+Config.getInstance().getModuleConfigByName("KeyFactory").getConfigFile()+" no encontrado.");
			}
		} catch (ConfigException e) {
			throw new KeyFactoryConfigException("Config Exception");
		}

		//Leemos y procesamos el fichero con la configuracion
		XMLkeyFactoryParser xmlkeyfactoryconfig = new XMLkeyFactoryParser(configuracion);
		xmlkeyfactoryconfig.procesaConfiguracion();
		logger.debug(configuracion.getAbsolutePath()+" parseado");

		numero_trazos = xmlkeyfactoryconfig.getNumerotrazos();
		key_params_configurations = xmlkeyfactoryconfig.getKey_params_configurations();

	}
	public void reloadConfig() throws KeyFactoryConfigException{
		File configuracion;
		try {
```

```
                configuracion = new
File(Config.getInstance().getModuleConfigByName("KeyFactory").getConfigFile());
                if(!configuracion.exists()){
                    throw new KeyFactoryConfigException("Fichero:
"+Config.getInstance().getModuleConfigByName("KeyFactory").getConfigFile()+" no encontrado.");
                }
            } catch (ConfigException e) {
                throw new KeyFactoryConfigException("Config Exception");
            }

            //Leemos y procesamos el fichero con la configuracion
            XMLkeyFactoryParser xmlkeyfactoryconfig = new XMLkeyFactoryParser(configuracion);
            xmlkeyfactoryconfig.procesaConfiguracion();
            logger.debug(configuracion.getAbsolutePath()+" parseado");

            numero_trazos = xmlkeyfactoryconfig.getNumerotrazos();
            key_params_configurations = xmlkeyfactoryconfig.getKey_params_configurations();
        }

        public int getNumero_trazos() {
            return numero_trazos;
        }

        public void setNumero_trazos(int numero_trazos) {
            this.numero_trazos = numero_trazos;
        }

        public ArrayList<KeyFactoryParameterConfigBean> getKey_params_configurations() {
            return key_params_configurations;
        }

        public void setKey_params_configurations(ArrayList<KeyFactoryParameterConfigBean>
key_params_configurations) {
            this.key_params_configurations = key_params_configurations;
        }

        public KeyFactoryParameterConfigBean getKeyFactoryParameterConfigBeanByName(String param){
            for(KeyFactoryParameterConfigBean paramAux : key_params_configurations){
                if(paramAux.getName().equals(param)){
                    return paramAux;
                }
            }
            return null;
        }
        public Key createKey(ArrayList<Trazo> trazos){
            logger.debug("Iniciando fabricación de Llave");
            long inicioTs = Calendar.getInstance().getTimeInMillis();
            long min_value = 0;
            long max_value = 0;
            long avg_value = 0;
            ArrayList<iKeyParameter> array_parametros = new ArrayList<iKeyParameter>();
            Key key = new Key();
            // para cada parametro
            for(KeyFactoryParameterConfigBean keyParamAux : key_params_configurations){
                //Reseteo contadores
                min_value = 0;
                max_value = 0;
                avg_value = 0;
                if(keyParamAux.isEnabled()){
                    for(Trazo trazo : trazos){
                        try {
                            Object parametro = Class.forName(keyParamAux.getClassName()).newInstance();
                            Method m = parametro.getClass().getDeclaredMethod("quantize", Trazo.class);
                            Long value = (Long)m.invoke(parametro,trazo);

                            //fijo el valor minimo
                            if( (min_value == 0) || (value.longValue()<min_value)){
```

```
                    min_value = value.longValue();
                }
                if( (max_value == 0) || (value.longValue()>min_value)){
                    max_value = value.longValue();
                }
                avg_value += value.longValue();


            } catch (InstantiationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalArgumentException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SecurityException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (NoSuchMethodException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        avg_value = avg_value/trazos.size();

        logger.info("Parametro<"+keyParamAux.getName()+">");
        logger.info("    Valor medio: "+avg_value);


        try {
            Object parametro = Class.forName(keyParamAux.getClassName()).newInstance();

            Method m = parametro.getClass().getDeclaredMethod("setParamName", String.class);
            m.invoke(parametro,keyParamAux.getName());

            m = parametro.getClass().getDeclaredMethod("setValue", Long.class);
            m.invoke(parametro,avg_value);

            m = parametro.getClass().getDeclaredMethod("setTolerance", Integer.class);
            m.invoke(parametro,0);

            array_parametros.add((iKeyParameter)parametro);

        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    } catch (InvocationTargetException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }


                }
            }
            key.setParametros(array_parametros);
            long finTs = Calendar.getInstance().getTimeInMillis();
            logger.debug("Fin fabricación Llave ha tardado: "+(finTs-inicioTs)+" ms");
            return key;

        }

}
```

B.9.3. hda.auth.keys.KeyFactoryConfigException.java

```
package hda.auth.keys;

public class KeyFactoryConfigException extends Exception{

    private static final long serialVersionUID = 1L;

    public KeyFactoryConfigException() {
        super();
    }

    public KeyFactoryConfigException(String arg0, Throwable arg1) {
        super(arg0, arg1);
    }

    public KeyFactoryConfigException(String arg0) {
        super(arg0);
    }

    public KeyFactoryConfigException(Throwable arg0) {
        super(arg0);
    }

}
```

B.10. Paquete hda.auth.locks

B.10.1.hda.auth.locks.Lock.java

```
package hda.auth.locks;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import hda.auth.keys.Key;
import hda.auth.parameters.iKeyParameter;

public class Lock implements Serializable {

	private static final long serialVersionUID = 1L;
	private transient Logger logger = LogManager.getLogger(this.getClass().getName());
	private ArrayList<iKeyParameter> parametros;

	public ArrayList<iKeyParameter> getParametros() {
		return parametros;
	}

	public void setParametros(ArrayList<iKeyParameter> parametros) {
		this.parametros = parametros;
	}

	public boolean open(Key key){
		logger = LogManager.getLogger(this.getClass().getName());
		logger.info("<LOGIN>");
		logger.info("   <FECHA>");
		logger.info("      "+Calendar.getInstance().getTime().toString());
		logger.info("   </FECHA>");

		boolean resultado = false;
		if(key.getParametros().size()!=parametros.size()){
			logger.error("El numero de parametro de la llave no coincide con el cerrojo");
			return false;
		}
		for(iKeyParameter parametro_lock : getParametros()){
			for(iKeyParameter parametro_key : key.getParametros()){
				if(parametro_lock.getParamName().equals(parametro_key.getParamName())){
					//comprobamos si cumple
					double margen = (new Double(parametro_lock.getValue()) * ((new
Double(parametro_lock.getTolerance()))/new Double(100)));

					long valor_lock_min = parametro_lock.getValue() -  (long)margen;
					long valor_lock_max = parametro_lock.getValue() + (long)margen;
					long valor_key = parametro_key.getValue();
					logger.debug("   <CHECK_PARAM>");
					logger.debug("      <Parametro>"+parametro_key.getParamName()+"</Parametro>");
					logger.debug("      <Valor_Minimo>"+valor_lock_min+"</Valor_Minimo>");
					logger.debug("      <Valor_Maximo>"+valor_lock_max+"</Valor_Maximo>");
					logger.debug("      <Valor_Llave>"+valor_key+"</Valor_Llave>");
					if( (valor_key>=valor_lock_min) && (valor_key<=valor_lock_max) ){
						logger.debug("      <Resultado>OK</Resultado>");
						logger.debug("   </CHECK_PARAM>");
						resultado = true;
					}else{
						resultado = false;
						logger.debug("      <Resultado>ERROR</Resultado>");
```

```
                                logger.debug("   </CHECK_PARAM>");
                                logger.info("   <RESULTADO>"+resultado+"</RESULTADO>");
                                logger.info("</LOGIN>");
                                return resultado;
                            }
                        }
                    }
                }
                logger.info("   <RESULTADO>"+resultado+"</RESULTADO>");
                logger.info("</LOGIN>");
                return resultado;
            }

        }
```

B.10.2. hda.auth.locks.LockFactory.java

```
package hda.auth.locks;

import hda.auth.config.Config;
import hda.auth.config.ConfigException;
import hda.auth.config.bean.LockFactoryParameterConfigBean;
import hda.auth.config.parsers.XMLlockFactoryParser;
import hda.auth.img.Trazo;
import hda.auth.parameters.iKeyParameter;

import java.io.File;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class LockFactory {

	private final Logger logger = LogManager.getLogger(this.getClass().getName());

	private static LockFactory inst;
	private int numero_trazos = 1;
	private String dispositivo = "";
	private int horas_validez = 24;
	private ArrayList<LockFactoryParameterConfigBean> lock_params_configurations = new
ArrayList<LockFactoryParameterConfigBean>();


	public static LockFactory getInstance() throws LockFactoryConfigException{
		if(inst==null){
			inst = new LockFactory();
		}
		return inst;
	}

	public LockFactory() throws LockFactoryConfigException {
		File configuracion;
		try {
			configuracion = new
File(Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile());
			if(!configuracion.exists()){
				throw new LockFactoryConfigException("Fichero:
"+Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile()+" no encontrado.");
			}
		} catch (ConfigException e) {
			throw new LockFactoryConfigException("Config Exception");
		}

		//Leemos y procesamos el fichero con la configuracion
		XMLlockFactoryParser xmllockfactoryconfig = new XMLlockFactoryParser(configuracion);
		xmllockfactoryconfig.procesaConfiguracion();
		logger.debug(configuracion.getAbsolutePath()+" parseado");

		numero_trazos = xmllockfactoryconfig.getNumerotrazos();
		dispositivo = xmllockfactoryconfig.getDispositivo();
		horas_validez = xmllockfactoryconfig.getHorasvalidez();
		lock_params_configurations = xmllockfactoryconfig.getLock_params_configurations();

	}
	public void reloadConfig() throws LockFactoryConfigException {
		File configuracion;
```

```
            try {
                configuracion = new
File(Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile());
                if(!configuracion.exists()){
                    throw new LockFactoryConfigException("Fichero:
"+Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile()+" no encontrado.");
                }
            } catch (ConfigException e) {
                throw new LockFactoryConfigException("Config Exception");
            }

            //Leemos y procesamos el fichero con la configuracion
            XMLlockFactoryParser xmllockfactoryconfig = new XMLlockFactoryParser(configuracion);
            xmllockfactoryconfig.procesaConfiguracion();
            logger.debug(configuracion.getAbsolutePath()+" parseado");

            numero_trazos = xmllockfactoryconfig.getNumerotrazos();
            dispositivo = xmllockfactoryconfig.getDispositivo();
            horas_validez = xmllockfactoryconfig.getHorasvalidez();
            lock_params_configurations = xmllockfactoryconfig.getLock_params_configurations();
        }
        public void saveConfig() throws LockFactoryConfigException {
            File configuracion;
            try {
                configuracion = new
File(Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile());
                if(!configuracion.exists()){
                    throw new LockFactoryConfigException("Fichero:
"+Config.getInstance().getModuleConfigByName("LockFactory").getConfigFile()+" no encontrado.");
                }
            } catch (ConfigException e) {
                throw new LockFactoryConfigException("Config Exception");
            }

            //exportamos la configuración
            XMLlockFactoryParser xmllockfactoryconfig = new XMLlockFactoryParser(configuracion);
            xmllockfactoryconfig.guardaConfiguracion();
            logger.debug(configuracion.getAbsolutePath()+" guardado");

        }

        public int getNumero_trazos() {
            return numero_trazos;
        }

        public void setNumero_trazos(int numero_trazos) {
            this.numero_trazos = numero_trazos;
        }

        public String getDispositivo() {
            return dispositivo;
        }

        public void setDispositivo(String dispositivo) {
            this.dispositivo = dispositivo;
        }

        public int getHoras_validez() {
            return horas_validez;
        }

        public void setHoras_validez(int horas_validez) {
            this.horas_validez = horas_validez;
        }

        public ArrayList<LockFactoryParameterConfigBean> getLock_params_configurations() {
            return lock_params_configurations;
```

```
	}

	public void setLock_params_configurations(ArrayList<LockFactoryParameterConfigBean>
lock_params_configurations) {
		this.lock_params_configurations = lock_params_configurations;
	}

	public LockFactoryParameterConfigBean getLockFactoryParameterConfigBeanByName(String param){
		for(LockFactoryParameterConfigBean paramAux : lock_params_configurations){
			if(paramAux.getName().equals(param)){
				return paramAux;
			}
		}
		return null;
	}
	public Lock createLock(ArrayList<Trazo> trazos){
		logger.debug("Iniciando fabricación de Cerradura");
		long inicioTs = Calendar.getInstance().getTimeInMillis();
		long min_value = 0;
		long max_value = 0;
		long avg_value = 0;
		float tolerance = 0;
		ArrayList<iKeyParameter> array_parametros = new ArrayList<iKeyParameter>();
		Lock lock = new Lock();
		// para cada parametro
		for(LockFactoryParameterConfigBean lockParamAux : lock_params_configurations){
			//Reseteo contadores
			min_value = 0;
			max_value = 0;
			avg_value = 0;
			if(lockParamAux.isEnabled()){
				for(Trazo trazo : trazos){
					try {
						Object parametro = Class.forName(lockParamAux.getClassName()).newInstance();
						Method m = parametro.getClass().getDeclaredMethod("quantize", Trazo.class);
						Long value = (Long)m.invoke(parametro,trazo);

						//fijo el valor minimo
						if( (min_value == 0) || (value.longValue()<min_value)){
							min_value = value.longValue();
						}
						if( (max_value == 0) || (value.longValue()>min_value)){
							max_value = value.longValue();
						}
						avg_value += value.longValue();


					} catch (InstantiationException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (IllegalAccessException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (ClassNotFoundException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (IllegalArgumentException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (InvocationTargetException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (SecurityException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					} catch (NoSuchMethodException e) {
						// TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                    }
                }
                avg_value = avg_value/trazos.size();

                long diferencia_max_avg=max_value - avg_value;
                long diferencia_avg_min=avg_value - min_value;
                if(diferencia_max_avg>diferencia_avg_min){
                    tolerance = new Float(1)-( new Float(avg_value)/new Float(max_value)));
                }else{
                    tolerance = new Float(1)-( new Float(min_value)/new Float(avg_value)));
                }
                int toleranceInt = new Float(tolerance*100).intValue();
                logger.info("Parametro<"+lockParamAux.getName()+">");
                logger.info("    Valor medio: "+avg_value);
                logger.info("    Tolerancia calculada: "+toleranceInt+" %");
                logger.info("    Tolerancia incertidumbre:"+lockParamAux.getTolerance()+" %");
                logger.info("    Tolerancia Total:"+(toleranceInt+lockParamAux.getTolerance())+" %");
                toleranceInt = toleranceInt+lockParamAux.getTolerance();

                try {
                    Object parametro = Class.forName(lockParamAux.getClassName()).newInstance();

                    Method m = parametro.getClass().getDeclaredMethod("setParamName", String.class);
                    m.invoke(parametro,lockParamAux.getName());

                    m = parametro.getClass().getDeclaredMethod("setValue", Long.class);
                    m.invoke(parametro,avg_value);

                    m = parametro.getClass().getDeclaredMethod("setTolerance", Integer.class);
                    m.invoke(parametro,toleranceInt);

                    array_parametros.add((iKeyParameter)parametro);

                } catch (InstantiationException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (SecurityException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (NoSuchMethodException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalArgumentException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }


            }
        }

        lock.setParametros(array_parametros);
        long finTs = Calendar.getInstance().getTimeInMillis();
        logger.debug("Fin fabricación Cerradura ha tardado: "+(finTs-inicioTs)+" ms");
        return lock;

    }
```

```
}
```

B.10.3.hda.auth.locks.LockFactoryConfigException.java

```
package hda.auth.locks;

public class LockFactoryConfigException extends Exception {

	private static final long serialVersionUID = 1L;

	public LockFactoryConfigException() {
		super();
	}

	public LockFactoryConfigException(String arg0, Throwable arg1) {
		super(arg0, arg1);
	}

	public LockFactoryConfigException(String arg0) {
		super(arg0);
	}

	public LockFactoryConfigException(Throwable arg0) {
		super(arg0);
	}
}
```

B.11. Paquete hda.auth.parameters

B.11.1.hda.auth.parameters.HeightKeyParameter.java

```
package hda.auth.parameters;

import java.awt.event.MouseEvent;
import java.io.Serializable;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import hda.auth.img.Trazo;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;


public class HeightKeyParameter implements iKeyParameter, Serializable  {


	private static final long serialVersionUID = 1L;

	private transient Logger logger = LogManager.getLogger(this.getClass().getName());

	private String dispositivo;
	private String paramName;
	private Long value;
	private Integer tolerance;

	private int maxValue = -1;
	private int minValue = -1;

	@Override
	public String getParamName() {
		return paramName;
	}

	@Override
	public void setParamName(String paramName) {
		this.paramName = paramName;

	}

	@Override
	public Long getValue() {
		return value;
	}

	@Override
	public void setValue(Long value) {
		this.value = value;

	}

	@Override
	public Integer getTolerance() {
		return tolerance;
	}

	@Override
	public void setTolerance(Integer tolerance) {
		this.tolerance = tolerance;
```

```
    }

    @Override
    public Long quantize(Trazo trazo) throws KeyParameterException {
        try {
            dispositivo = LockFactory.getInstance().getDispositivo();
        } catch (LockFactoryConfigException e) {
            dispositivo = "udraw";
        }
        logger.debug("Iniciando quantización de HeightKeyParameter");
        long inicioTs = Calendar.getInstance().getTimeInMillis();
        if("udraw".equals(dispositivo)){
            for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                if(mouseEvent.getID()==MouseEvent.MOUSE_DRAGGED){
                    if(maxValue==-1){
                        maxValue = mouseEvent.getY();
                    }
                    if(minValue==-1){
                        minValue = mouseEvent.getY();
                    }
                    if(mouseEvent.getY() > maxValue){
                        maxValue = mouseEvent.getY();
                    }else if(mouseEvent.getY() < minValue) {
                        minValue = mouseEvent.getY();
                    }
                }
            }
        }else{
            for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                if(mouseEvent.getID()==MouseEvent.MOUSE_MOVED){
                    if(maxValue==-1){
                        maxValue = mouseEvent.getY();
                    }
                    if(minValue==-1){
                        minValue = mouseEvent.getY();
                    }
                    if(mouseEvent.getY() > maxValue){
                        maxValue = mouseEvent.getY();
                    }else if(mouseEvent.getY() < minValue) {
                        minValue = mouseEvent.getY();
                    }
                }
            }
        }

        long finTs = Calendar.getInstance().getTimeInMillis();
        logger.debug("Fin quantización de HeightKeyParameter ha tardado: "+(finTs-inicioTs)+" ms");
        return new Long(maxValue-minValue);
    }


}
```

B.11.2. hda.auth.parameters.HSenseChangeKeyParameter.java

```
package hda.auth.parameters;


import java.awt.event.MouseEvent;
import java.io.Serializable;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import hda.auth.img.Trazo;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;
import hda.auth.parameters.iKeyParameter;

public class HSenseChangeKeyParameter implements iKeyParameter, Serializable  {


	private static final long serialVersionUID = 1L;

	private transient Logger logger = LogManager.getLogger(this.getClass().getName());

	private String dispositivo;
	private String paramName;
	private Long value;
	private Integer tolerance;

	private int hsensechanges = 0;
	private boolean hcreciendo = true;

	@Override
	public String getParamName() {
		return paramName;
	}

	@Override
	public void setParamName(String paramName) {
		this.paramName = paramName;

	}

	@Override
	public Long getValue() {
		return value;
	}

	@Override
	public void setValue(Long value) {
		this.value = value;

	}

	@Override
	public Integer getTolerance() {
		return tolerance;
	}

	@Override
	public void setTolerance(Integer tolerance) {
		this.tolerance = tolerance;

	}
```

```
    @Override
    public Long quantize(Trazo trazo) throws KeyParameterException {
        try {
            dispositivo = LockFactory.getInstance().getDispositivo();
        } catch (LockFactoryConfigException e) {
            dispositivo = "udraw";
        }
        logger.debug("Iniciando quantización de HSenseChangeKeyParameter");
        long inicioTs = Calendar.getInstance().getTimeInMillis();
        int posicionXAnterior = 0;
        if("udraw".equals(dispositivo)){
            for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                if(mouseEvent.getID()==MouseEvent.MOUSE_DRAGGED){
                    if( (mouseEvent.getX() > posicionXAnterior) && (hcreciendo)){
                        hcreciendo = false;
                        hsensechanges++;
                    }else if( (mouseEvent.getX() < posicionXAnterior) && (!hcreciendo)){
                        hcreciendo = true;
                        hsensechanges++;
                    }
                    posicionXAnterior = mouseEvent.getX();
                }
            }
        }else{
            for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                if(mouseEvent.getID()==MouseEvent.MOUSE_MOVED){
                    if( (mouseEvent.getX() > posicionXAnterior) && (hcreciendo)){
                        hcreciendo = false;
                        hsensechanges++;
                    }else if( (mouseEvent.getX() < posicionXAnterior) && (!hcreciendo)){
                        hcreciendo = true;
                        hsensechanges++;
                    }
                    posicionXAnterior = mouseEvent.getX();
                }
            }
        }

        long finTs = Calendar.getInstance().getTimeInMillis();
        logger.debug("Fin quantización de HSenseChangeKeyParam ha tardado: "+(finTs-inicioTs)+" ms");

        return new Long(hsensechanges);
    }


}
```

B.11.3. hda.auth.parameters.iKeyParameter.java

```
package hda.auth.parameters;

import hda.auth.img.Trazo;


public interface iKeyParameter {
	public String getParamName();
	public void setParamName(String paramName);
	public Long getValue();
	public void setValue(Long value);
	public Integer getTolerance();
	public void setTolerance(Integer tolerance);
	/**
	 * Método que cuantiza el parámetro en un valor de tipo long
	 *
	 * @param trazo clase {@link Trazo} que modela el trazo realizado sobre los que efectuar el cálculo
	 * @return valor que representa el parámetro
	 * @exception KeyParameterException
	 */
	public Long quantize(Trazo trazo) throws KeyParameterException;


}
```

B.11.4.hda.auth.parameters.KeyParameterException.java

```
package hda.auth.parameters;

import java.io.Serializable;

public class KeyParameterException extends Exception implements Serializable  {

	private static final long serialVersionUID = 1L;

	public KeyParameterException() {
		super();
		// TODO Auto-generated constructor stub
	}

	public KeyParameterException(String arg0, Throwable arg1) {
		super(arg0, arg1);
		// TODO Auto-generated constructor stub
	}

	public KeyParameterException(String arg0) {
		super(arg0);
		// TODO Auto-generated constructor stub
	}

	public KeyParameterException(Throwable arg0) {
		super(arg0);
		// TODO Auto-generated constructor stub
	}

}
```

B.11.5. hda.auth.parameters.PixelDensityKeyParameter.java

```
package hda.auth.parameters;

import hda.auth.img.Trazo;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.Serializable;
import java.util.Calendar;

import javax.imageio.ImageIO;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


public class PixelDensityKeyParameter implements iKeyParameter, Serializable  {


	private static final long serialVersionUID = 1L;

	private transient Logger logger = LogManager.getLogger(this.getClass().getName());

	private String paramName;
	private Long value;
	private Integer tolerance;

	@Override
	public String getParamName() {
		return paramName;
	}

	@Override
	public void setParamName(String paramName) {
		this.paramName = paramName;
	}

	@Override
	public Long getValue() {
		return value;
	}

	@Override
	public void setValue(Long value) {
		this.value = value;
	}

	@Override
	public Integer getTolerance() {
		return tolerance;
	}

	@Override
	public void setTolerance(Integer tolerance) {
		this.tolerance = tolerance;
	}


	@Override
	public Long quantize(Trazo trazo) throws KeyParameterException {
		logger.debug("Iniciando quantización de PixelDensityKeyParam");
		long inicioTs = Calendar.getInstance().getTimeInMillis();
		long numPixeles = 0;
	try {
```

```
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
                ImageIO.write(trazo.getImagen(), "bmp", baos);
                byte[] imagenBytes = baos.toByteArray();

            for(int i=100;i<imagenBytes.length;i++){
                if(imagenBytes[i]!=0x00){
                    numPixeles++;
                }
            }
            } catch (IOException e) {
                throw new KeyParameterException(e.getMessage());
            }
            long finTs = Calendar.getInstance().getTimeInMillis();
            logger.debug("Fin quantización de PixelDensityKeyParam ha tardado: "+(finTs-inicioTs)+" ms");
            return new Long(numPixeles);
        }

}
```

B.11.6. hda.auth.parameters.VSenseChangeKeyParameter.java

```
package hda.auth.parameters;


import java.awt.event.MouseEvent;
import java.io.Serializable;
import java.util.Calendar;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import hda.auth.img.Trazo;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;
import hda.auth.parameters.iKeyParameter;

public class VSenseChangeKeyParameter implements iKeyParameter, Serializable  {

	private static final long serialVersionUID = 1L;

	private transient Logger logger = LogManager.getLogger(this.getClass().getName());

	private String dispositivo;
	private String paramName;
	private Long value;
	private Integer tolerance;

	private int vsensechanges = 0;
	private boolean vcreciendo = true;

	@Override
	public String getParamName() {
		return paramName;
	}

	@Override
	public void setParamName(String paramName) {
		this.paramName = paramName;

	}

	@Override
	public Long getValue() {
		return value;
	}

	@Override
	public void setValue(Long value) {
		this.value = value;

	}

	@Override
	public Integer getTolerance() {
		return tolerance;
	}

	@Override
	public void setTolerance(Integer tolerance) {
		this.tolerance = tolerance;

	}

	@Override
```

```
	public Long quantize(Trazo trazo) throws KeyParameterException {
		try {
			dispositivo = LockFactory.getInstance().getDispositivo();
		} catch (LockFactoryConfigException e) {
			dispositivo = "udraw";
		}
		logger.debug("Iniciando quantización de VSenseChangeKeyParam");
		long inicioTs = Calendar.getInstance().getTimeInMillis();
		int posicionYAnterior = 0;
		if("udraw".equals(dispositivo)){
			for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
				if(mouseEvent.getID()==MouseEvent.MOUSE_DRAGGED){
					if( (mouseEvent.getY() < posicionYAnterior) && (vcreciendo)){
						vcreciendo = false;
						vsensechanges++;
					}else if( (mouseEvent.getY() > posicionYAnterior) && (!vcreciendo)){
						vcreciendo = true;
						vsensechanges++;
					}
					posicionYAnterior = mouseEvent.getY();
				}
			}
		}else{
			for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
				if(mouseEvent.getID()==MouseEvent.MOUSE_MOVED){
					if( (mouseEvent.getY() < posicionYAnterior) && (vcreciendo)){
						vcreciendo = false;
						vsensechanges++;
					}else if( (mouseEvent.getY() > posicionYAnterior) && (!vcreciendo)){
						vcreciendo = true;
						vsensechanges++;
					}
					posicionYAnterior = mouseEvent.getY();
				}
			}
		}

		long finTs = Calendar.getInstance().getTimeInMillis();
		logger.debug("Fin quantización de VSenseChangeKeyParam ha tardado: "+(finTs-inicioTs)+" ms");

		return new Long(vsensechanges);
	}


}
```

B.11.7.hda.auth.parameters.WidthKeyParameter.java

```
package hda.auth.parameters;

import java.awt.event.MouseEvent;
import java.io.Serializable;
import java.util.Calendar;

import hda.auth.img.Trazo;
import hda.auth.locks.LockFactory;
import hda.auth.locks.LockFactoryConfigException;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


public class WidthKeyParameter implements iKeyParameter, Serializable {

	private static final long serialVersionUID = 1L;

	private transient Logger logger = LogManager.getLogger(this.getClass().getName());

	private String dispositivo;
	private String paramName;
	private Long value;
	private Integer tolerance;

	private int maxValue = -1;
	private int minValue = -1;

	@Override
	public String getParamName() {
		return paramName;
	}

	@Override
	public void setParamName(String paramName) {
		this.paramName = paramName;

	}

	@Override
	public Long getValue() {
		return value;
	}

	@Override
	public void setValue(Long value) {
		this.value = value;

	}

	@Override
	public Integer getTolerance() {
		return tolerance;
	}

	@Override
	public void setTolerance(Integer tolerance) {
		this.tolerance = tolerance;

	}

	@Override
	public Long quantize(Trazo trazo) throws KeyParameterException {
```

```
            try {
                dispositivo = LockFactory.getInstance().getDispositivo();
            } catch (LockFactoryConfigException e) {
                dispositivo = "udraw";
            }
            logger.debug("Iniciando quantización de WidthKeyParameter");
            long inicioTs = Calendar.getInstance().getTimeInMillis();
            if("udraw".equals(dispositivo)){
                for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                    if(mouseEvent.getID()==MouseEvent.MOUSE_DRAGGED){
                        if(maxValue==-1){
                            maxValue = mouseEvent.getX();
                        }
                        if(minValue==-1){
                            minValue = mouseEvent.getX();
                        }
                        if(mouseEvent.getX() > maxValue){
                            maxValue = mouseEvent.getX();
                        }else if(mouseEvent.getX() < minValue) {
                            minValue = mouseEvent.getX();
                        }
                    }
                }
            }else{
                for(MouseEvent mouseEvent : trazo.getMouseEvents() ){
                    if(mouseEvent.getID()==MouseEvent.MOUSE_MOVED){
                        if(maxValue==-1){
                            maxValue = mouseEvent.getX();
                        }
                        if(minValue==-1){
                            minValue = mouseEvent.getX();
                        }
                        if(mouseEvent.getX() > maxValue){
                            maxValue = mouseEvent.getX();
                        }else if(mouseEvent.getX() < minValue) {
                            minValue = mouseEvent.getX();
                        }
                    }
                }
            }

            long finTs = Calendar.getInstance().getTimeInMillis();
            logger.debug("Fin quantización de WidthKeyParameter ha tardado: "+(finTs-inicioTs)+" ms");
            return new Long(maxValue-minValue);
        }

    }
```

B.12. Scripts Batch

B.12.1.ejecuta.bat

```
echo Iniciando Aplicacion Autenticacion

start java -cp lib/ExternalParam.jar;lib/bcprov-ext-jdk15on-148.jar;lib/Filters.jar;lib/forms-1.1.0.jar;lib/jcalendar-1.3.3.jar;lib/looks-2.0.1.jar;lib/looks-2.1.4.jar;lib/swing-layout-1.0.3.jar;lib/TableLayout.jar;lib/log4j-core-2.0-beta9.jar;lib/log4j-api-2.0-beta9.jar;bin -Dmainconfig="config/mainconfig.xml" hda.auth.gui.PrincipalGUI

echo Lanzando Driver uDraw...

.\driver_tablet\uDrawTablet.7.1.0\bin\uDrawTablet.exe
```

B.12.2.ejecuta_sinDriver.bat

```
java -cp lib/ExternalParam.jar;lib/bcprov-ext-jdk15on-148.jar;lib/Filters.jar;lib/forms-1.1.0.jar;lib/jcalendar-1.3.3.jar;lib/looks-2.0.1.jar;lib/looks-2.1.4.jar;lib/swing-layout-1.0.3.jar;lib/TableLayout.jar;lib/log4j-core-2.0-beta9.jar;lib/log4j-api-2.0-beta9.jar;bin -Dmainconfig="config/mainconfig.xml" hda.auth.gui.PrincipalGUI
```

C. Código PFG_EXTRAS

C.1. Paquete hda.auth.cifrado

C.1.1. hda.auth.cifrado.GeneradorClave.java

```
package hda.auth.cifrado;


import java.util.ArrayList;


public class GeneradorClave{
	public static String generaClaveSecretaUID(){

		ArrayList<String> lista = new ArrayList<String>();

		try{
			lista.add(SystemUniqueId.getBiosSerialNumber());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		try{
			lista.add(SystemUniqueId.getCpuProcessorId());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		try{
			lista.add(SystemUniqueId.getDiskDriveSerialNumber());
		}catch(Exception e){
			lista.add("DEFAULT");
		} catch (Throwable e) {
			lista.add("DEFAULT");
		}
		//aquí se podría complicar el algoritmo si quisieramos

		String entropia = "";
		String passwordHashStr = "";
		for(String cadena:lista){
			entropia = SecurityToolBox.getSHA256(cadena+entropia);
		}
		passwordHashStr = entropia;
		return passwordHashStr;

	}

}
```

C.1.2. hda.auth.cifrado.SecurityToolBox.java

```
package hda.auth.cifrado;

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.bouncycastle.crypto.BufferedAsymmetricBlockCipher;
import org.bouncycastle.crypto.DataLengthException;
import org.bouncycastle.crypto.digests.SHA256Digest;
import org.bouncycastle.crypto.encodings.PKCS1Encoding;
import org.bouncycastle.crypto.engines.AESEngine;
import org.bouncycastle.crypto.engines.RSAEngine;
import org.bouncycastle.crypto.modes.CBCBlockCipher;
import org.bouncycastle.crypto.paddings.PKCS7Padding;
import org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher;
import org.bouncycastle.crypto.params.KeyParameter;
import org.bouncycastle.crypto.params.RSAKeyParameters;
import org.bouncycastle.crypto.params.RSAPrivateCrtKeyParameters;
import org.bouncycastle.crypto.util.PrivateKeyFactory;
import org.bouncycastle.crypto.util.PublicKeyFactory;
import org.bouncycastle.util.encoders.Hex;

public class SecurityToolBox {

	public static String getSHA256(String texto){
	   SHA256Digest  digest = new SHA256Digest();
	   byte[]  hash = new byte[digest.getDigestSize()];
	   byte[]  bytesTexto = texto.getBytes();
	   digest.update(bytesTexto, 0, bytesTexto.length);
	   digest.doFinal(hash, 0);
	   String hashStr = new String(Hex.encode(hash));
	   return hashStr;
	}

	public static void cifrarConClavePrivada(InputStream is, OutputStream os, byte[] key) throws Exception {

		BufferedAsymmetricBlockCipher encryptCipher;
		byte[] buf = new byte[16];              //input buffer
		byte[] obuf = new byte[512];            //output buffer
		System.out.println(" CIFRAR-->");
		try {
			RSAPrivateCrtKeyParameters privateKey = (RSAPrivateCrtKeyParameters)
PrivateKeyFactory.createKey(key);
			encryptCipher = new BufferedAsymmetricBlockCipher (new PKCS1Encoding(new RSAEngine()));
			encryptCipher.init(true, privateKey);

			int total = is.available();
			ByteArrayOutputStream bos = new ByteArrayOutputStream();
			int noBytesRead = 0;       //number of bytes read from input
			int len = encryptCipher.getInputBlockSize();
			buf = new byte[len];
			while ((noBytesRead = is.read(buf)) >= 0) {
				try{
					encryptCipher.processBytes(buf, 0, noBytesRead);
					obuf = encryptCipher.doFinal();
					bos.write(obuf, 0, obuf.length);
					encryptCipher.reset();
				}catch(DataLengthException dle){
					obuf = encryptCipher.doFinal();
					bos.write(obuf, 0, noBytesRead);
				}
				float bosFloat = bos.size();
				float totalFloat = total;
```

```
                    float porcentaje = (bosFloat/totalFloat)*100;
                    System.out.println(porcentaje);
                }
                os.write(bos.toByteArray());
            }catch(Exception e){
                e.printStackTrace();
            }

        }
        public static void descifrarConClavePublica(InputStream is, OutputStream os, byte[] key) throws Exception {

            BufferedAsymmetricBlockCipher decryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            System.out.println(" DESCIFRAR-->");
            try {
                RSAKeyParameters publicKey = (RSAKeyParameters) PublicKeyFactory.createKey(key);
                decryptCipher = new BufferedAsymmetricBlockCipher (new PKCS1Encoding(new RSAEngine()));
                decryptCipher.init(false, publicKey);


                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                int noBytesRead = 0;       //number of bytes read from input
                int len = decryptCipher.getInputBlockSize();
                buf = new byte[len];

                while ((noBytesRead = is.read(buf)) >= 0) {
                    try{
                        decryptCipher.processBytes(buf, 0, noBytesRead);
                        obuf = decryptCipher.doFinal();
                        bos.write(obuf, 0, obuf.length);
                        decryptCipher.reset();
                    }catch(DataLengthException dle){
                        obuf = decryptCipher.doFinal();
                        bos.write(obuf, 0, noBytesRead);
                    }
                    //float porcentaje = (bos.size()/total)*100;
                    System.out.println(bos.size());
                }
                os.write(bos.toByteArray());
            }catch(Exception e){
                e.printStackTrace();
            }

        }

        public static void cifrarConClaveSecreta(InputStream is, OutputStream os, byte[] key) throws Exception{

            PaddedBufferedBlockCipher encryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            System.out.println(" CIFRAR-->");
            //encryptCipher = new PaddedBufferedBlockCipher(new AESEngine());
            encryptCipher = new PaddedBufferedBlockCipher(new CBCBlockCipher(new AESEngine()), new
PKCS7Padding());
            encryptCipher.init(true, new KeyParameter(key));
            try {
                int noBytesRead = 0;       //number of bytes read from input
                int noBytesProcessed = 0;   //number of bytes processed
                while ((noBytesRead = is.read(buf)) >= 0) {
                    //System.out.println(noBytesRead +" bytes read");
                    noBytesProcessed = encryptCipher.processBytes(buf, 0, noBytesRead, obuf, 0);
                    os.write(obuf, 0, noBytesProcessed);
                }

                //System.out.println(noBytesRead +" bytes read");
                noBytesProcessed = encryptCipher.doFinal(obuf, 0);
```

```
                //System.out.println(noBytesProcessed +" bytes processed");
                os.write(obuf, 0, noBytesProcessed);

                os.flush();
            }
            catch (java.io.IOException e) {
                System.out.println(e.getMessage());
            }
        }

        public static void descifrarConClaveSecreta(InputStream is, OutputStream os, byte[] key) throws Exception {

            PaddedBufferedBlockCipher decryptCipher;
            byte[] buf = new byte[16];              //input buffer
            byte[] obuf = new byte[512];            //output buffer
            System.out.println(" DESCIFRAR-->");
            //decryptCipher =  new PaddedBufferedBlockCipher(new AESEngine());
            decryptCipher =  new PaddedBufferedBlockCipher(new CBCBlockCipher(new AESEngine()),new
PKCS7Padding());
            decryptCipher.init(false, new KeyParameter(key));
            try {
                // Bytes read from in will be decrypted
                // Read in the decrypted bytes from in InputStream and and
                // write them in cleartext to out OutputStream
                int noBytesRead = 0;        //number of bytes read from input
                int noBytesProcessed = 0;   //number of bytes processed
                while ((noBytesRead = is.read(buf)) >= 0) {
                    //System.out.println(noBytesRead +" bytes read");
                    noBytesProcessed = decryptCipher.processBytes(buf, 0, noBytesRead, obuf, 0);
                    os.write(obuf, 0, noBytesProcessed);
                }
                //System.out.println(noBytesRead +" bytes read");
                noBytesProcessed = decryptCipher.doFinal(obuf, 0);
                //System.out.println(noBytesProcessed +" bytes processed");
                os.write(obuf, 0, noBytesProcessed);
                os.flush();
            }
            catch (java.io.IOException e) {
                System.out.println(e.getMessage());
            }
        }

}
```

C.1.3. hda.auth.cifrado.SystemUniqueId.java

```
package hda.auth.cifrado;


import java.util.Scanner;

public class SystemUniqueId {

   public static String getBiosSerialNumber() throws Throwable {
      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "bios", "get", "serialNumber" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;
   }
   public static String getDiskDriveSerialNumber() throws Throwable {

      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "diskdrive", "get", "serialnumber" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;

   }
   public static String getCpuProcessorId() throws Throwable {
      Process process = Runtime.getRuntime().exec(new String[] { "wmic", "cpu", "get", "processorid" });
      process.getOutputStream().close();
      Scanner sc = new Scanner(process.getInputStream());
      String respuesta = "";
      while(sc.hasNext()){
            respuesta = respuesta.concat(sc.next());
      }
      return respuesta;
      //SerialNumber: CZC7150HB7
      //ProcessorId: BFEBFBFF00000F65
   }
}
```

C.2. Paquete hda.auth.gui

C.2.1. hda.auth.gui.FrmCheckLogin.java

```
package hda.auth.gui;



import hda.auth.gui.listeners.CheckLoginListener;

import java.io.Serializable;
import java.security.Security;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class FrmCheckLogin extends JFrame implements Serializable{

	private static final long serialVersionUID = 1L;
	private static FrmCheckLogin inst;
	private JFrame framePadre;
	private JPanel panelFormulario;

	private JButton selectLoginBoton;
	private JTextField selectLoginField;
	private JButton selectPubKeyBoton;
	private JTextField selectPubKeyField;

	private JButton comprobarBoton;

	{
		//Set Look & Feel
		try {
			javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
		} catch(Exception e) {
			System.out.println("ERROR LOOK&FEEL");
		}
	}

	public static void main(String[] args) {
		try {
			Security.addProvider(new BouncyCastleProvider());
			SwingUtilities.invokeLater(new Runnable() {
				public void run() {
					inst = new FrmCheckLogin();
					inst.setLocationRelativeTo(null);
					inst.setVisible(true);
				}
			});
		} catch (Exception e) {
			System.exit(ERROR);
		}

	}
```

```
    public FrmCheckLogin(){
        initComponents();
    }

    private void initComponents(){
        //setSize(450,250);
        //setLocation((framePadre.getX())+(framePadre.getWidth()/2)-(getWidth()/2) ,
(framePadre.getY())+(framePadre.getHeight()/2)-(getHeight()/2));
        //setResizable(false);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle("COMPROBADOR TOKEN DE LOGIN");
        CheckLoginListener checkLoginListener = new CheckLoginListener(this);
        addWindowListener(checkLoginListener);

        panelFormulario = new JPanel();
        FormLayout checkLoginLayout = new FormLayout("max(p;100), max(p;100), max(p;100),
max(p;100)","max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30)");
        panelFormulario.setLayout(checkLoginLayout);
        panelFormulario.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

        {
            selectLoginField = new JTextField(JTextField.WHEN_FOCUSED);
            panelFormulario.add(selectLoginField, new CellConstraints(1, 2, 3, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            selectLoginBoton = new JButton();
            selectLoginBoton.setName("Seleccionar Token");
            selectLoginBoton.setText("Seleccionar Token");
            selectLoginBoton.setActionCommand("Seleccionar Token");
            selectLoginBoton.addActionListener(checkLoginListener);
            panelFormulario.add(selectLoginBoton, new CellConstraints(4, 2, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));

        }
        {
            selectPubKeyField = new JTextField();
            panelFormulario.add(selectPubKeyField, new CellConstraints(1, 3, 3, 1, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        {
            selectPubKeyBoton = new JButton();
            selectPubKeyBoton.setName("Seleccionar PubKey");
            selectPubKeyBoton.setText("Seleccionar PubKey");
            selectPubKeyBoton.setActionCommand("Seleccionar PubKey");
            selectPubKeyBoton.addActionListener(checkLoginListener);
            panelFormulario.add(selectPubKeyBoton, new CellConstraints(4, 3, 1, 1, CellConstraints.FILL,
CellConstraints.CENTER));

        }

        {
            comprobarBoton = new JButton();
            comprobarBoton.setName("Comprobar");
            comprobarBoton.setText("Comprobar");
            comprobarBoton.setActionCommand("Comprobar");
            comprobarBoton.addActionListener(checkLoginListener);
            panelFormulario.add(comprobarBoton, new CellConstraints(1, 5, 4, 2, CellConstraints.FILL,
CellConstraints.CENTER));
        }
        add(panelFormulario);
        pack();
        setResizable(false);

    }
```

```
	public JFrame getFramePadre() {
		return framePadre;
	}

	public void setFramePadre(JFrame framePadre) {
		this.framePadre = framePadre;
	}

	public JPanel getPanelFormulario() {
		return panelFormulario;
	}

	public void setPanelFormulario(JPanel panelFormulario) {
		this.panelFormulario = panelFormulario;
	}
	public JButton getSelectLoginBoton() {
		return selectLoginBoton;
	}
	public void setSelectLoginBoton(JButton selectLoginBoton) {
		this.selectLoginBoton = selectLoginBoton;
	}
	public JTextField getSelectLoginField() {
		return selectLoginField;
	}
	public void setSelectLoginField(JTextField selectLoginField) {
		this.selectLoginField = selectLoginField;
	}
	public JButton getSelectPubKeyBoton() {
		return selectPubKeyBoton;
	}
	public void setSelectPubKeyBoton(JButton selectPubKeyBoton) {
		this.selectPubKeyBoton = selectPubKeyBoton;
	}
	public JTextField getSelectPubKeyField() {
		return selectPubKeyField;
	}
	public void setSelectPubKeyField(JTextField selectPubKeyField) {
		this.selectPubKeyField = selectPubKeyField;
	}
	public JButton getComprobarBoton() {
		return comprobarBoton;
	}
	public void setComprobarBoton(JButton comprobarBoton) {
		this.comprobarBoton = comprobarBoton;
	}

}
```

C.2.2. hda.auth.gui.FrmPassword.java

```
package hda.auth.gui;



import hda.auth.gui.listeners.PasswordListener;

import java.io.Serializable;
import java.security.Security;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class FrmPassword extends JFrame implements Serializable{

	private static final long serialVersionUID = 1L;
	private static FrmPassword inst;
	private JFrame framePadre;
	private JPanel panelFormulario;
	private JLabel claveAccesoLabel;
	private JTextArea claveAccesoArea;

	private JButton generarBoton;

	{
		//Set Look & Feel
		try {
			javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
		} catch(Exception e) {
			System.out.println("ERROR LOOK&FEEL");
		}
	}

	public static void main(String[] args) {
		try {
			Security.addProvider(new BouncyCastleProvider());
			SwingUtilities.invokeLater(new Runnable() {
				public void run() {
					inst = new FrmPassword();
					inst.setLocationRelativeTo(null);
					inst.setVisible(true);
				}
			});
		} catch (Exception e) {
			System.exit(ERROR);
		}

	}
	public FrmPassword(){
		initComponents();
	}

	private void initComponents(){
		setSize(450,250);
```

```
            //setLocation((framePadre.getX())+(framePadre.getWidth()/2)-(getWidth()/2) ,
(framePadre.getY())+(framePadre.getHeight()/2)-(getHeight()/2));
            //setResizable(false);
            setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            setTitle("GENERADOR PASSWORD");
            PasswordListener passwordListener = new PasswordListener(this);
            addWindowListener(passwordListener);

            panelFormulario = new JPanel();
            FormLayout adminPasswordLayout = new FormLayout("max(p;140), max(p;100), max(p;100),
max(p;100)","max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30), max(p;30)");
            panelFormulario.setLayout(adminPasswordLayout);
            panelFormulario.setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

            {
                  claveAccesoLabel = new JLabel();
                  claveAccesoLabel.setText("Clave acceso");
                  claveAccesoLabel.setForeground(new java.awt.Color(0,64,128));
                  claveAccesoLabel.setFont(new java.awt.Font("Arial Black",0,16));
                  panelFormulario.add(claveAccesoLabel, new CellConstraints(1, 3, 1, 2, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            {
                  claveAccesoArea = new JTextArea();
                  claveAccesoArea.setLineWrap(true);
                  claveAccesoArea.setName("password");
                  panelFormulario.add(claveAccesoArea, new CellConstraints(2, 3, 2, 2, CellConstraints.FILL,
CellConstraints.CENTER));

            }

            {
                  generarBoton = new JButton();
                  generarBoton.setName("Generar");
                  generarBoton.setText("Generar");
                  generarBoton.setActionCommand("Generar");
                  generarBoton.addActionListener(passwordListener);
                  panelFormulario.add(generarBoton, new CellConstraints(2, 6, 2, 1, CellConstraints.FILL,
CellConstraints.CENTER));
            }
            add(panelFormulario);

      }

      public JFrame getFramePadre() {
            return framePadre;
      }

      public void setFramePadre(JFrame framePadre) {
            this.framePadre = framePadre;
      }

      public JPanel getPanelFormulario() {
            return panelFormulario;
      }

      public void setPanelFormulario(JPanel panelFormulario) {
            this.panelFormulario = panelFormulario;
      }

      public JLabel getClaveAccesoLabel() {
            return claveAccesoLabel;
      }

      public void setClaveAccesoLabel(JLabel claveAccesoLabel) {
            this.claveAccesoLabel = claveAccesoLabel;
```

```
    }

    public JTextArea getClaveAccesoArea() {
        return claveAccesoArea;
    }

    public void setClaveAccesoField(JTextArea claveAccesoArea) {
        this.claveAccesoArea = claveAccesoArea;
    }

    public JButton getGenerarBoton() {
        return generarBoton;
    }

    public void setGenerarBoton(JButton generarBoton) {
        this.generarBoton = generarBoton;
    }

}
```

C.3. Paquete hda.auth.gui.listeners

C.3.1. hda.auth.gui.listeners.CheckLoginListener.java

```
package hda.auth.gui.listeners;

import hda.auth.cifrado.SecurityToolBox;
import hda.auth.gui.FrmCheckLogin;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Date;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.bouncycastle.util.encoders.Hex;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class CheckLoginListener implements WindowListener, ActionListener{
	FrmCheckLogin frmCheckLogin;
	String textoFirma = "";
	public CheckLoginListener(FrmCheckLogin frmCheckLogin){
		this.frmCheckLogin = frmCheckLogin;
	}

	public void actionPerformed(ActionEvent evento) {


		//EVENTO SELECCIONAR TOKEN
		if(evento.getActionCommand().equals("Seleccionar Token")){
			String directorioRaiz = ".";
			JFileChooser explorador = new JFileChooser(directorioRaiz+File.separator);
			explorador.setDialogTitle("Seleccionar Token...");
			//explorador.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
			String[] extensiones = {"xml"};
			explorador.setFileFilter(new FileNameExtensionFilter("HDA Token",extensiones));

			int seleccion = explorador.showOpenDialog(frmCheckLogin);
			switch(seleccion) {
			case JFileChooser.APPROVE_OPTION:
				frmCheckLogin.getSelectLoginField().setText(explorador.getSelectedFile().getAbsolutePath());
				break;

			case JFileChooser.CANCEL_OPTION:
				frmCheckLogin.getSelectLoginField().setBackground(new Color(200,100,100));
				break;
```

```
                case JFileChooser.ERROR_OPTION:
                    frmCheckLogin.getSelectLoginField().setBackground(new Color(200,100,100));
                    break;
                }

            }
            //EVENTO SELECCIONAR CLAVE PUBLICA
            if(evento.getActionCommand().equals("Seleccionar PubKey")){
                String directorioRaiz = ".";
                JFileChooser explorador = new JFileChooser(directorioRaiz+File.separator);
                explorador.setDialogTitle("Seleccionar Clave Publica...");
                //explorador.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                String[] extensiones = {"key"};
                explorador.setFileFilter(new FileNameExtensionFilter("HDA key",extensiones));

                int seleccion = explorador.showOpenDialog(frmCheckLogin);
                switch(seleccion) {
                case JFileChooser.APPROVE_OPTION:

    frmCheckLogin.getSelectPubKeyField().setText(explorador.getSelectedFile().getAbsolutePath());
                    break;

                case JFileChooser.CANCEL_OPTION:
                    frmCheckLogin.getSelectPubKeyField().setBackground(new Color(200,100,100));
                    break;

                case JFileChooser.ERROR_OPTION:
                    frmCheckLogin.getSelectPubKeyField().setBackground(new Color(200,100,100));
                    break;
                }
            }
            //EVENTO GENERAR COMPROBAR TOKEN
            if(evento.getActionCommand().equals("Comprobar")){
                textoFirma = "";
                boolean continuarComprobacion = true;
                if( (frmCheckLogin.getSelectLoginField().getText()==null) ||
("".equals(frmCheckLogin.getSelectLoginField().getText())) ){
                    frmCheckLogin.getSelectLoginField().setBackground(new Color(200,100,100));
                    continuarComprobacion = false;
                }
                if( (frmCheckLogin.getSelectPubKeyField().getText()==null) ||
("".equals(frmCheckLogin.getSelectPubKeyField().getText())) ){
                    frmCheckLogin.getSelectPubKeyField().setBackground(new Color(200,100,100));
                    continuarComprobacion = false;
                }
                if(continuarComprobacion){
                    String cadenaComprobacion = "";
                    //Aqui parseamos el fichero del token y comprobamos la firma
                    try{
                        File fXmlFile = new File(frmCheckLogin.getSelectLoginField().getText());
                        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
                        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                        Document doc = dBuilder.parse(fXmlFile);
                        doc.getDocumentElement().normalize();
                        System.out.println("Root element:" + doc.getDocumentElement().getNodeName());

                        if (doc.hasChildNodes()) {
                            cadenaComprobacion = printNote(doc.getChildNodes(),cadenaComprobacion);

                        }
                    } catch (ParserConfigurationException pce) {
                        pce.printStackTrace();
                    } catch (SAXException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
```

```
                    JOptionPane.showMessageDialog (null, cadenaComprobacion, "RESULTADO
COMPROBACION", JOptionPane.INFORMATION_MESSAGE);
                }

            }


        }

        private String printNote(NodeList nodeList, String cadenaComprobacion) {
            for (int count = 0; count < nodeList.getLength(); count++) {

                Node tempNode = nodeList.item(count);

                if (tempNode.getNodeType() == Node.ELEMENT_NODE) {

                    // get node name and value
                    System.out.println("\nNode Name =" + tempNode.getNodeName() + " [OPEN]");
                    System.out.println("Node Value =" + tempNode.getTextContent());

                    if("user".equals(tempNode.getNodeName())){
                        cadenaComprobacion +=
tempNode.getNodeName()+":"+tempNode.getFirstChild().getTextContent()+"\n";
                        textoFirma += tempNode.getFirstChild().getTextContent();
                    } else if("noAntes".equals(tempNode.getNodeName())){
                        cadenaComprobacion += tempNode.getNodeName()+":"+new
Date(Long.parseLong(tempNode.getFirstChild().getTextContent())).toString()+"\n";
                        textoFirma += tempNode.getFirstChild().getTextContent();
                    }else if("noDespues".equals(tempNode.getNodeName())){
                        cadenaComprobacion += tempNode.getNodeName()+":"+new
Date(Long.parseLong(tempNode.getFirstChild().getTextContent())).toString()+"\n";
                        textoFirma += tempNode.getFirstChild().getTextContent();
                    }else if("firma".equals(tempNode.getNodeName())){
                        String hashStrCalculado = SecurityToolBox.getSHA256(textoFirma);
                        ByteArrayOutputStream baos = new ByteArrayOutputStream();
                        FileInputStream fis = null;
                        BufferedInputStream bais = null;
                        try {
                            int bytesLeidos;
                            byte[] buffer = new byte[1024];
                            fis = new FileInputStream(frmCheckLogin.getSelectPubKeyField().getText());
                            bais = new BufferedInputStream(fis);

                            while( (bytesLeidos = bais.read(buffer)) > 0 ){
                                baos.write(buffer,0,bytesLeidos);
                                baos.flush();
                            }
                            ByteArrayInputStream bis = new
ByteArrayInputStream(Hex.decode(tempNode.getFirstChild().getTextContent()));
                            ByteArrayOutputStream bos = new ByteArrayOutputStream();
                            SecurityToolBox.descifrarConClavePublica(bis, bos,
Hex.decode((baos.toByteArray())));
                            String hashStrRecuperado = new String(bos.toByteArray());
                            if(hashStrCalculado.equals(hashStrRecuperado)){
                                cadenaComprobacion += "FIRMA VALIDA";
                            }else{
                                cadenaComprobacion += "FIRMA NO VALIDA";
                            }
                        } catch (IOException e) {
                            e.printStackTrace();
                        } catch (Exception e) {
                            e.printStackTrace();
                        } finally{
                            try {
                                if(bais!=null){
                                    bais.close();
```

```
                                    }
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                            }

                        }
                        if (tempNode.hasChildNodes()) {
                            // loop again if has child nodes
                            cadenaComprobacion = printNote(tempNode.getChildNodes(),cadenaComprobacion);
                        }
                        System.out.println("Node Name =" + tempNode.getNodeName() + " [CLOSE]");

                    }
                }
                return cadenaComprobacion;
            }
            @Override
            public void windowActivated(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
            @Override
            public void windowClosed(WindowEvent arg0) {
                // TODO Auto-generated method stub
                frmCheckLogin.getFramePadre().setEnabled(true);
            }
            @Override
            public void windowClosing(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
            @Override
            public void windowDeactivated(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
            @Override
            public void windowDeiconified(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
            @Override
            public void windowIconified(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
            @Override
            public void windowOpened(WindowEvent arg0) {
                // TODO Auto-generated method stub
            }
        }
```

C.3.2. hda.auth.gui.listeners.PasswordListener.java

```
package hda.auth.gui.listeners;

import hda.auth.cifrado.GeneradorClave;
import hda.auth.gui.FrmPassword;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

public class PasswordListener implements WindowListener, ActionListener{
	FrmPassword frmPassword;
	public PasswordListener(FrmPassword frmPassword){
		this.frmPassword = frmPassword;
	}
	public void actionPerformed(ActionEvent evento) {
		//EVENTO GENERAR PASSWORD ADMIN
		if(evento.getActionCommand().equals("Generar")){
			frmPassword.getClaveAccesoArea().setText(GeneradorClave.generaClaveSecretaUID());
		}
	}
	@Override
	public void windowActivated(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}
	@Override
	public void windowClosed(WindowEvent arg0) {
		// TODO Auto-generated method stub
		frmPassword.getFramePadre().setEnabled(true);
	}
	@Override
	public void windowClosing(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}
	@Override
	public void windowDeactivated(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}
	@Override
	public void windowDeiconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}
	@Override
	public void windowIconified(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}
	@Override
	public void windowOpened(WindowEvent arg0) {
		// TODO Auto-generated method stub

	}

}
```

C.4. Scripts Batch

C.4.1. ejecuta_checkToken.bat

```
java -cp lib/bcprov-ext-jdk15on-148.jar;lib/forms-1.1.0.jar;lib/jcalendar-1.3.3.jar;lib/looks-2.0.1.jar;lib/looks-2.1.4.jar;lib/swing-layout-1.0.3.jar;lib/TableLayout.jar;lib/log4j-core-2.0-beta9.jar;lib/log4j-api-2.0-beta9.jar;bin hda.auth.gui.FrmCheckLogin
```

C.4.2. ejecuta_generador.bat

```
java -cp lib/bcprov-ext-jdk15on-148.jar;lib/forms-1.1.0.jar;lib/jcalendar-1.3.3.jar;lib/looks-2.0.1.jar;lib/looks-2.1.4.jar;lib/swing-layout-1.0.3.jar;lib/TableLayout.jar;lib/log4j-core-2.0-beta9.jar;lib/log4j-api-2.0-beta9.jar;bin hda.auth.gui.FrmPassword
```